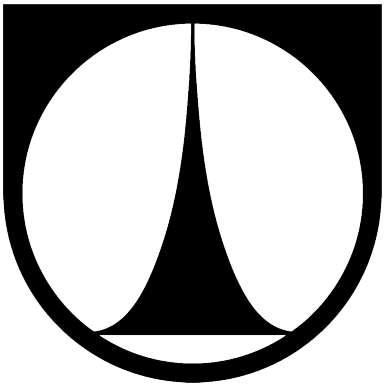
TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií



DIPLOMOVÁ PRÁCE

V Liberci, 18. května 2013 **Bc. Jakub Ponikelský**

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektronika a informatika

Studijní obor: 1802T007 – Informační technologie

Ovládání zabezpečovací kamery EYE-02 protokolem XMPP

Control of security camera EYE-02 via XMPP protocol

Bc. Jakub Ponikelský

Vedoucí práce: doc. RNDr. Pavel Satrapa, Ph.D.

Konzultant: Ing. Jan Halama, JABLOCOM s. r. o.

Pracoviště: Ústav nových technologií a aplikované informatiky

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum:

18. května 2013

Podpis:

. . . . . .

Bc. Jakub Ponikelský

Poděkování

Touto cestou bych rád poděkoval vedoucímu diplomové práce doc. RNDr. Pavlu Satrapovi, Ph.D., za jeho cenné připomínky při vytváření a dokončování práce. Rád bych také poděkoval společnosti JABLOCOM s. r. o. za zprostředkování práce a jmenovitě pak Ing. Janu Halamovi a Ing. Pavlu Vikovi, Ph.D., za rady a veškerou pomoc při konzultacích.

Dále bych velmi rád poděkoval své rodině za oporu, kterou mi poskytovala během celého studia, konkrétně pak za morální podporu během kompletování této diplomové práce a vytvoření potřebného zázemí pro klidné studium na vysoké škole.

Použitý software

Tato práce byla vysázena pomocí systému LaTeX (prosředí Eclipse Juno ve verzi 4.2.1, plugin TeXlipse a sázecí systém MiKTeX ve verzi 2.9) a programu Software Ideas Modeler (vývojové a další diagramy) pod operačním systémem Microsoft Windows 7 Professional 64b. Jako nástroj pro tvorbu zdrojových kódů bylo použito Microsoft Visual Studio 2010 Ultimate.

Kontakt

E-mail: jakub.ponikelsky@tul.cz

Abstrakt

Cílem této diplomové práce je rozšířit množinu komunikačních kanálů sloužících pro ovládání bezpečnostní kamery EYE-02 o protokol XMPP. V teoretické části je stručně představen zadavatel projektu společnost Jablocom s. r. o. a základní charakteristické vlastnosti jejího produktu – kamery EYE-02. Dále je zde seznámení se základními mechanismy protokolu XMPP, jejichž pochopení je důležité k vybudování služby, která by měla nahrazovat XMPP klienta pro vybranou kameru.

V praktické části je nejdříve popsán výběr XMPP serveru a jeho nastavení tak, aby byla zaručena maximální úroveň zabezpečení přenášených dat ale i samotného systému. Dále je zde navržena podoba databáze, která tvoří prostředníka mezi službou a existující infrastrukturou společnosti Jablocom s. r. o. K této databázi je pak navržena a implementována WCF služba pro přidávání povelů do databáze a naopak získávání a filtrování požadavků v databázi.

Následně je zde navržena a implementována služba, která umožňuje přenášet uživatelské zprávy (poplachy, stavové zprávy, chyby) z kamery na IM klienta uživatele a naopak od něj přebírá ovládací povely a zobrazuje aktuální stav kamery (watch, sleep). Služba má také implementovány základní postupy pro přenos souborů a videohovorů prostřednictvím protokolu XMPP. Dále je pak implementována hlavní služba, která kontroluje správnost chodu jednotlivých vláken a v periodických intervalech spouští nebo naopak ukončuje vlákna vybraných kamer. Poslední vytvořenou aplikací je pak webová stránka sloužící k administraci celého systému.

Klíčová slova:

XPPP protokol,
bezpečnostní kamera,
přenos zpráv,
XMPP server,
XMPP služba

Abstract

The objective of this thesis is to add XMPP protocol into a set of communication channels used to control security camera EYE-02. The theoretical part presents the project submitter Jablocom ltd. and basic properties of the product – EYE-02. Then there is the introduction to the basic mechanisms of the protocol XMPP, whose understanding is important to build a service that would replace XMPP client for the selected camera.

The practical part of the thesis describes selection of the XMPP server and configuration of the XMPP server to ensure the safety of transmitted data as well as the system itself. Then there is a database designed, which is an intermediary between the service and the infrastructure of Jablocom ltd. and WCF Service for adding commands to the database and retrieving and filtering commands from the database.

Then there is a service designed and implemented. The service allows to transmit messages (alarms, status messages, errors) from the camera to the IM client of user and to take control command from the user and to display the current status of the camera (watch, sleep). The service has also implemented the basic procedures for transferring files and making video calls via XMPP protocol. There is also main service implemented. Main service checks the runtime of individual fibers in periodic intervals and launches or terminates fibers of selected cameras. Last created application is a webpage that is used to administer the whole system.

Keywords:

XMPP protocol,
security camera,
instant messaging,
XMPP server,
XMPP service

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

AGPL	Affero General Public License
AOL	America Online
API	Application Programming Interface
ASP	Active Server Pages
CD	Compact Disc
CNN	Cable News Network
DB	Database/Databáze
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
GPL	GNU General Public License
GSM	Global System for Mobile Communications, Groupe Spécial Mobile
GUI	Graphical user interface
HSQLDB	Hyper Structured Query Language Database
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICQ	I seek you
IETF	Internet Engineering Task Force
IM	Instant Messaging
IP	Internet Protocol
IRC	Internet Relay Chat
JDK	Java Development Kit
JID	Jabber Identifier
MIT	Massachusetts Institute of Technology
MMS	Multimedia Messaging Service
MSN	Microsoft Network
MS SQL	Microsoft SQL Server
MVC	Model-view-controller
ODBC	Open Database Connectivity
PI sensor	Passive infrared sensor
RFC	Request for Comments
RF module	Radio Frequency Module
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SASL	Simple Authentication and Security Layer
SIP	Session Initiation Protocol
SMS	Short Message Service

SQL	Structured Query Language
SSL	Secure Sockets Layer
$STUN$	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
USB	Universal Serial Bus
UTF	UCS Transformation Format
$VOIP$	Voice over Internet Protocol
WCF	Windows Communication Foundation
XEP	XMPP Extension Protocol
XML	Extensible Markup Language
$XMPP$	Extensible Messaging and Presence Protocol
XSF	XMPP Standard Foundation

1 Úvod

Komunikace je jednou z nejdůležitějších součástí lidského života. Je na ní závislé úplně vše, od uspokojování těch nejzákladnějších lidských potřeb (např. obstarávání potravy), přes naše oblíbené činnosti či zabezpečení našich majetků a životů, až po disciplíny důležité pro budoucí vývoj lidstva (vzdělávání, výzkum aj.). Proto se komunikací neboli přenosem informací lidstvo aktivně zabývá a snaží se tuto disciplínu co nejvíce vylepšovat a přizpůsobovat ji svým požadavkům. Výsledkem těchto snah jsou nové komunikační kanály a to jak ty dnes již dávno překonané (kouřové signály, domluvené mávání atd.), tak ty relativně moderní (telegraf, telefon).

S nástupem počítačů a internetu se lidstvu otevřela nebývalá možnost komunikace. Původně sice bylo možné předávat informace pouze v podobě textových souborů, ale lidé velmi brzy potenciál internetu objevili a tak začaly vznikat kanály, které se přímo zaměřovaly na komunikaci mezi lidmi. Jedním z nejstarších zástupců je například elektronická pošta, decentralizovaný systém pro předávání zpráv, který dnes používá snad každý, kdo vlastní počítač s přístupem do internetu. Ovšem i elektronická pošta má své nevýhody, jako je např. nevyžádaná pošta či podvrhnutí zprávy třetí stranou a navíc je zde stále problém, že je celá služba takzvaně offline. Tedy uživatel odešle zprávu a neví, zda si ji uživatel na druhé straně přečte ihned či až za nějakou dobu, tedy zdali je druhý uživatel právě u počítače.

Reakcí na tento problém jsou takzvané IRC sítě, jedná se o jednu z prvních variant komunikace v reálném čase. Znamená to, že co jeden uživatel napíše, druhý vidí ihned na svém počítači (prostřednictvím prohlížeče nebo přímo IRC klienta). Ovšem i IRC má své nedostatky. Zejména nemožnost směrování zprávy, jedinou variantou je využít některou z funkcí IRC a vytvořit si takzvaně soukromý kanál dvou uživatelů, ve kterém si tak mohou tito dva uživatelé svobodně komunikovat. IRC pro to má nezbytné prostředky, jako je správa oprávnění jednotlivých uživatelů (moderované kanály, heslované kanály či tzv. ban pro určitého uživatele v určitém kanálu).

Výsledkem snah o to, aby měl každý uživatel svou soukromou schránku do které je mu možné adresovat zprávu a zároveň komunikace v reálném čase je instant messaging. V případě IM má každý uživatel jedinečnou adresu, která je unikátní v celém internetu a také má možnost vybrané adresy ukládat do svého seznamu (buddylist, roster apod.). Ovšem v čem je oproti svým předchůdcům unikátní je práce s aktuálním stavem uživatele tzv. presence. Díky tomu může uživatel IM, dříve než odešle svou

zprávu, zjistit, zdali je druhý uživatel právě u svého počítače či zda není aktuálně zaneprázdněn apod. I IM má své nedostatky, mezi které patří například nevyžádané zprávy či nekompatibilita různých IM protokolů. I proto vznikla a stále vzniká celá řada různých IM, například: ICQ, AOL instant messenger, Yahoo! messenger, MSN messenger či Jabber/XMPP.

IM lze využít i k jiným věcem než je promluva dvou a více uživatelů. V době, kdy je internet stále více také průmyslovým nástrojem, lze IM využít k poskytování některých druhů služeb. Například mohou být uživateli zasílány aktuální výsledky zápasů jeho oblíbeného sportovního týmu či aktuální informace o počasí ve vybrané oblasti apod.

Toto je i motivací vzniku této práce, tedy spojit veřejně dostupný kanál pro přenos textových zpráv, který používá velká skupina lidí, a bezpečnost jeho uživatele či majetku tohoto uživatele. Společnost Jablocom s. r. o. se zaměřuje na řešení v oblasti GSM technologií a jedním z jejích produktů je i zabezpečovací kamera s názvem EYE-02. V této práci tak bude navžena a implementována služba, která umožní, aby si uživatel přidal svou bezpečnostní kameru do svého již existujícího IM klienta a ta ho informovala o svém aktuálním stavu a stavu svého okolí.

Stejně tak bude moci uživatel kameru ovládat pomocí příkazů v podobě odeslaných textových zpráv. Důležité je také brát na zřetel charakter zařízení, tedy maximalizovat snahu o bezpečnost odesílaných informací a také snahu o ověření doručení zprávy od kamery ke klientovi.

Prostřednictvím IM klientů lze v dnešní době také odesílat soubory a využívat přímého video spojení. Proto by bylo dobré diskutovat i tuto oblast tak, aby uživatel mohl přímo ve svém IM klientu vidět v reálném čase, co se děje v okolí jeho kamery, nebo mohl obdržet fotografii z doby, kdy došlo k narušení střeženého prostoru.

2 Seznámení s prostředím

V následujících kapitolách dojde k představení prostředí, ve kterém bude služba vyvíjena. Bude představena společnost Jablocom s. r. o. s jejím produktem EYE-02 (2.1) a také použitý IM kanál XMPP (2.2).

2.1 Společnost Jablocom a její produkty

Hlavní zájmovou oblastí firmy Jablocom s. r. o. je výroba a dodávání GSM telekomunikačních a zabezpečovacích zařízení. Vznik společnosti se datuje do roku 2004, kdy byla z rodinné společnosti Jablotron vyčleněna malá skupina konstruktérů, která se zaměřila na inovace a využívání netradičních řešení v oblasti zabezpečení pod názvem Aplikovaný výzkum. Zkušenosti, které skupina získávala při vývoji GSM komunikátorů pro ovládání zabezpečovacích systémů, vyvrcholily v inovativní projekt stolního GSM telefonu. První model, jenž byl označen jako GDP-02, byl poprvé oficiálně představen na světové výstavě elektroniky, která se uskutečnila v říjnu roku 2004 v Hongkongu. Převratná novinka sklidila obrovský komerční úspěch, a to i díky sérii reportáží, které byly odvysílány televizní stanicí CNN, a tak společnost získala první objednávku od operátora GSM sítě ještě před koncem roku 2004. Vzhledem k tomuto úspěchu a také odlišnosti oproti hlavní zájmové oblasti firmy Jablotron bylo rozhodnuto o vytvoření dceřiné společnosti Jablocom s. r. o., zabývající se výhradně oborem telekomunikačních systémů. [1]

V současnosti je hlavní aktivitou firmy stále prodej a vývoj v oblasti aplikované elektroniky s důrazem na překvapivá a neotřelá řešení, využívání nejmodernějších cenově dostupných technologií, jednoduchost uživatelské obsluhy a vysokou kvalitu dodávaných produktů. Společnost Jablocom nabízí čtyři hlavní produktové řady: [1]

- EYE-02 / Eye See – GSM bezpečnostní kamera – zabezpečovací zařízení ve tvaru malé kamery obsahující řadu senzorů s možností rozšíření o další bezdrátové detektory. Velkou výhodou kamery je schopnost informovat uživatele o nastalé situaci užitím několika různých komunikačních kanálů.
- BlueSynergy – Bluetooth stolní telefon – tvarem je podobný JabloPhone, ale funguje jiným způsobem. BlueSynergy se prostřednictvím bluetooth automaticky spáruje s mobilním telefonem uživatele a výsledkem je, že je uživatel dostupný stále pod stejným mobilním číslem, ale využívá přednosti pevné linky.
- JabloPhone – GSM stolní telefon – běžný mobilní telefon komunikující přes GSM síť s designem telefonu stolního. Je vybaven displejem, QWERTY klávesnicí,

reproduktorem, tlačítky rychlého vytáčení či internetovým připojením. Telefon podporuje také pokročilé kancelářské funkce, jako jsou přesměrování hovorů, konferenční hovory či nadstandardní správa kontaktů.

- JabloPhone kits – nástavba k JabloPhone, která nabízí radiofrekvenční spojení s dalšími zařízeními. Uživatel tak například jedním stiskem tlačítka na svém řetízku či hodinkách přivolá lékařskou pomoc či může vzdáleně ovládat svoje domácí spotřebiče a předejít tak škodám nebo si jen zapnout topení dříve než dojde domů.

2.1.1 Kamera EYE-02

Kamera EYE-02 je bezpečnostní GSM kamera, poskytující kompletní ochranu střeženého prostoru v jednom zařízení. Důkazem tohoto tvrzení je několik získaných ocenění (např. Global Mobile Awards 2011). Mezi základní vlastnosti kamery EYE-02 patří: [1]

- Pět základních senzorů – senzor PIR (detekuje tepelné vyzařování např. lidského těla), akustický senzor (nadprůměrný hluk v oblasti kamery), senzor tříštění skla (rozpoznává tento specifický zvuk), senzor pohybu (zaznamenává pohyb v obraze) a senzor náklonu a umístění (ochrana proti manipulaci s kamerou).
- Bezdrátový přístup – kamera je vybavena GSM, RF modulem a záložní baterií.
- Paměťová karta – možnost uchovávat data po dlouhou dobu přímo v kameře.
- GSM modul – kamera je v podstatě mobilní telefon, umí telefonovat, posílat a přijímat SMS i MMS nebo přistupovat k internetu a odesílat tak data na server či posílat e-maily.
- RF modul – možnost rozšíření kamery o další zařízení (dálkové ovládání, detektory kouře atd.).
- Noční vidění – kamera je schopna nahrávat video i v noci či temných oblastech.

Kamera EYE-02 má několik základních režimů. Prvním z nich je režim Odjištěno (Sleep) – jedná se o klidový stav kamery, ve kterém nesnímá hlídanou oblast. V tomto režimu jsou aktivovány pouze detektory bránící nežádoucí manipulaci s kamerou. Dále režim Zajištěno (Watch) – v tomto režimu jsou aktivovány všechny senzory kamery a kamera tak střeží hlídanou oblast. Režim Nastavení slouží ke změně vlastností kamery a jeho součástí jsou další režimy: Učení (přidání nového zařízení či přidání telefonního čísla do kamery), Test (slouží k ověření funkčnosti všech detektorů bez vyhlášení nežádoucích poplachových zpráv), Report (odešle uměle vytvořenou

zprávu na všechny kontakty spárované s kamerou) a USB (stav kamery, kdy se chová jako standardní USB disk). [1]

Kameru lze ovládat a nastavovat několika způsoby. Prvním je ovládání z počítače uživatele, k tomu slouží webové rozhraní Jablotool (dostupné na webové stránce www.jablotool.com). Toto rozhraní poskytuje několik základních aplikací (některé jsou zpoplatněny): [1]

- Access & Back-up – prohlížení událostí z kamery bez nutnosti přímého přístupu přímo na Jablotool serveru, kam jsou události automaticky zaznamenávány.
- Picture Link – hlášení zasílaná uživateli, obsahují odkaz na obrazovou informaci.
- Messenger Service – veškerá hlášení (SMS, MMS, e-maily) jsou odesílána Jablotool serverem a k jejich příjmu tak stačí pouze datový tarif.
- Watch Dog – kamera zasílá pravidelná hlášení, zda je v pořádku.
- Web Camera – snímky z kamery lze pomocí vytvořeného API zobrazit na webu.
- Live Streaming – možnost sledovat video z kamery přímo v prohlížeči.
- Timers – automatická změna stavu kamery v závislosti na čase (zajištění o víkendech, odjištění na začátku pracovní doby apod.).
- Flexi Limit – nastavení měsíčního limitu SMS, MMS či e-mailů, které kamera odešle.

Tyto funkce lze ovládat také pomocí desktopové aplikace, která je k dispozici na CD umístěném v balení kamery EYE-02. Další možností ovládání kamery je zadávání příkazů pomocí SMS. Kamera pomocí SMS informuje uživatele o svém stavu, zasílá obrázky z kamery (MMS), zasílá reporty o posledních několika událostech, zobrazuje informace o aktuální výši kreditu a uživatel naopak může přepínat stavy Zajištěno a Odjištěno, přidávat kontakty do kamery, měnit jazyk kamery apod. [1]

Další možností je kameru ovládat pomocí telefonních hovorů. Pokud je telefonní číslo, které na kameru volá, uvedeno v jejím seznamu kontaktů, je hovor přijat a přehraje se základní sada funkcí: poslech z mikrofonu kamery, vyžádání MMS s aktuálním obrázkem, vyžádání MMS poslední události typu Poplach, změna stavu kamery (Zajištěno/Odjištěno) atd. Novinkou je pak ovládání kamery pomocí mobilní aplikace, která je vytvořena pomocí jazyka HTML 5 a je tak dostupná pro mobilní zařízení se všemi operačními systémy (iOS, Windows Phone či Android). I tato aplikace umožňuje základní ovládání kamery – příjem obrázků, změnu stavu kamery či příjem poplachových zpráv. [1]

2.2 XMP protokol (XMPP)

The Extensible Messaging and Presence Protocol ve zkratce XMPP je otevřená technologie, jejímž primárním zaměřením je komunikace v reálném čase. XMPP je tudíž základním stavebním kamenem mnoha aplikací (např. Google Talk, Pidgin, Miranda, Trillian a mnoho dalších.), které v současnosti využívají miliony uživatelů po celém světě, a to nejen ke komunikaci s přáteli či rodinou, ale také například pro vnitrofiremní komunikaci. Všechny tyto aplikace velmi často spojují následující termíny: [2]

- Instant messaging (IM) – velmi rychlé doručování textových zpráv od odesilatele k adresátovi prostřednictvím sítě internet, tzv. komunikace v reálném čase.
- Informace o dostupnosti – indikátor toho, jestli je osoba se kterou by se mělo komunikovat schopná či ochotná komunikovat, tzv. status.
- Multi-party chat – vzájemná komunikace mezi více jak dvěma uživateli, tzv. konference či skupinová komunikace.
- Hlasové a videohovory – tedy přímá komunikace mezi uživateli pomocí hlasu případně i zraku.
- Groupware – možnost spolupráce (komunikace, odesílání souborů ...) mezi několika lidmi za účelem dosažení společného cíle.
- Middleware – spojení několika různých aplikací v jeden větší celek, například uživatel využívající Pidgin je schopen komunikovat i s uživateli využívající jinou aplikaci či jiný protokol.
- Syndikátor obsahu – uživatel nemusí kontrolovat, kdo a kdy mu zaslal zprávu, aplikace ho na to sama upozorní.
- Odesílání dat ve formátu XML – interní komunikace mezi klientem a serverem či serverem a serverem probíhá ve formátu XML.

Samotná technologie XMPP, které se v minulosti říkalo Jabber, byla vytvořena již v roce 1998 vývojářem jménem Jeremie Miller (ke spuštění prvního oficiálního XMPP serveru ovšem došlo až na začátku roku 1999). Technologie slavila úspěch a tak vznikla její vlastní komunita (Jabber open-source community), která ji v letech 1999–2000 nadále zdokonalovala. V letech 2002–2003 tak byla technologie oficiálně zformulována komunitou IETF a v roce 2004 vzniklo několik RFC týkajících se XMPP (RFC 3920, RFC 3921 a další). [2]

V současné době je standard XMPP udržován nezávislou a neziskovou organizací XMPP Standard Foundation (XSF), která definuje další standardy v oblasti IM, signalizace statusu a obecně komunikace v reálném čase nebo také nabízí informace a infrastrukturu pro všechny osoby zajímající se o XMPP (vývojáře, poskytovatele i koncové uživatele). [2]

Všechny standardy týkající se XMPP jsou aktuálně definovány v několika různých formách: [2]

- RFC, vydané IETF, popisující jádro protokolu XMPP (RFC 6120, nahrazující RFC 3920, definující jádro XMPP, RFC 6121, nahrazující RFC 3921, definující IM a stavovou informaci a RFC 6122, definující formát XMPP adresy).
- Několik dalších doplňujících RFC (Definice URI pro XMPP, šifrování XMPP ...)
- Internet-Drafts (internetové koncepty), které popisují další možnosti rozšíření XMPP technologie. Ty jsou navrhované třetími stranami (skupinami IETF či nezávislými uživateli XMPP) a schvalovány samotnou IETF.
- Standardy týkající jádra XMPP, které jsou vydávané XSF ve formě takzvaných XMPP Extension Protocol (XEP).

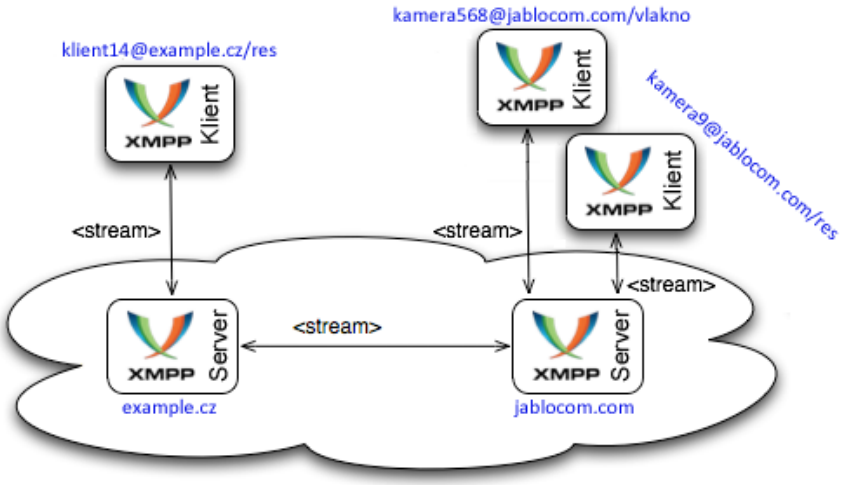
2.2.1 Vlastnosti XMPP

Jak bylo zmíněno dříve, XMPP protokol je technologie podporující instant messaging, uživatelskou stavovou informaci a další funkce, který je spravován XMPP Standard Foundation (dříve Jabber Software Foundation). Technologie byla vytvořena jako reakce na tehdejší konkurenční protokoly (ICQ, Microsoft Messenger ...), které byly uzavřené a uživatele mnohdy „zasypávaly“ nežádoucí reklamou. Vzhledem k těmto faktům XMPP nabízí tyto charakteristické vlastnosti: [2, 3]

- Klient-server – XMPP klienti (uživatelé) spolu nekomunikují přímo (výjimkou jsou speciální případy), ale prostřednictvím serveru, který zprostředkovává komunikaci a také udržuje seznam kontaktů a další informace o uživatelích.
- Decentralizovanost – XMPP razí podobnou cestu jako služba e-mail, tedy nevyužívá jeden server, který by obstarával všechny požadavky. Namísto toho má každý uživatel svůj jedinečný identifikátor, jehož prostřednictvím jsou mu směrovány XMPP pakety přes jeho domovský XMPP server (viz obrázek 1). To je výhodné zejména ve chvílích, kdy není dostupná jeho IP adresa, nebo se IP adresa mění (například při používání mobilního telefonu či tabletu).

Díky tomuto principu si může doslova každý vytvořit svůj vlastní XMPP server, což je velmi výhodné zejména pro firmy, které tak mohou firemní komunikaci libovolně upravovat na míru svým požadavkům.

Technologie XMPP standardně komunikuje prostřednictvím TCP spojení na portech 5222 a 5223 (klient – server) a 5269 (server – server).



Obrázek 1: Princip XMPP přenosu

- Bezpečnost – XMPP server může být izolovaný od vnější sítě a jeho prostřednictvím pak lze komunikovat pouze uvnitř privátní sítě, což přináší efektivní bezpečnostní mechanismus pro firemní intranet. V XMPP jsou definovány bezpečnostní mechanismy SASL či TLS, které tak umožňují zlepšit bezpečnost přenášených dat. Výhodou je také široká základna vývojářů aktivně pracujících na XMPP, díky které vznikají další a další bezpečnostní mechanismy.
- Otevřenost – XMPP je veřejně dostupný bezplatně, ve formě která je dobře čitelná pro vývojáře. Výsledkem je mnoho implementovaných knihoven, klientů a serverů.
- Standardizovanost – Jádro XMPP protokolu založené na dotazech v XML formě je popsáno v dokumentech RFC, rozšířená funkcionalita pak v dokumentech XEP.
- Osvědčenost – Od roku 1998 tuto technologii vylepšují stovky vývojářů, v síti Internet existují tisíce XMPP serverů a miliony uživatelů využívají XMPP služby jako je například Google Talk.
- Rozšiřitelnost a flexibilita – Standardizovány jsou pouze přenosové mechanismy, takže si kdokoli může funkcionalitu rozšířit o další požadované funkce. Některé

jsou již popsány v rozšířeních XEP (sdílení souborů, přenos videa, signalizace zpráv atd.). Neznamená to ovšem, že si kdokoli nemůže upravit funkcionalitu podle své potřeby.

- Rozmanitost – Jestliže si vývojář nechce vytvářet přímo své vlastní řešení, existuje velké množství veřejně dostupných implementací XMPP serverů či klientů.

V následujících kapitolách bude blíže představen obsah RFC dokumentů popisujících jádro XMPP technologie. Toto je důležité k zjištění jakým způsobem jsou formátovány XML dotazy odesílané a přijímané při požadavku uživatele o odeslání zprávy, změnu informace o jeho aktuální dostupnosti apod.

2.2.2 Adresa XMPP (JID)

XMPP entitou neboli členem komunikačního procesu může být každý prvek, který je možné v síti adresovat a podporuje zmíněnou XMPP technologii. Z historických důvodů se adresa entity označuje jako JID (Jabber Identifier). Validní JID je pak řetězec Unicode znaků, kódovaný pomocí UTF-8, který splňuje některé další definice (typicky odvozené od „stringprep“ – RFC 3454 nebo „nameprep“ – RFC 3491) a skládá se ze tří částí (localpart, domainpart a resourcepart). [4]

Domainpart je jediná část, která je povinná pro všechny JID, a to znamená, že i samotná domainpart je funkční JID. Domainpart typicky identifikuje domácí server, tedy server ke kterému se klienti připojují kvůli odesílání XML požadavků i dalších dat, ale může představovat i některé další služby. Důležitou podmínkou na domainpart je, aby byla FQDN (tedy řetězec rozlišitelný pomocí DNS), IP adresa verze 4, IP adresa verze 6 nebo minimálně řetězec rozlišitelný v lokální síti. [4]

Localpart je nepovinná součást adresy vložená před domainpart (oddělena „@“). Typicky identifikuje entitu, která se připojuje k serveru, ale může také představovat název místnosti ve službě pro hromadnou komunikaci. Další nepovinná část XMPP adresy je resourcepart, ta je umístěna za domainpart (oddělena „/“) a slouží k rozšíření (výsledná adresa je označována full JID) možností základní adresy ve formátu „localpart@domainpart“ (bare JID). Typicky identifikuje specifické připojení, díky čemuž může být uživatel připojen v jednom okamžiku z více zařízení (každé z nich je odlišeno jinou resourcepart) nebo objekt asociovaný s localpart (nick uživatele v místnosti pro hromadnou komunikaci). Resourcepart může obsahovat i znaky „/“ a „@“, ovšem je třeba brát na zřetel, že skládání znaků „/“ v XMPP nepředstavuje hierarchické uspořádání jako například v HTTP. [4]

Žádná ze zmíněných položek nesmí být být prázdná (řetězec délky nula) a také její délka nesmí přesáhnout 1023 bytů, což znamená, že maximální délka XMPP adresy je 3071 bytů (včetně znaků „@“ a „/“). Důležité je také podotknout, že JID je jediná adresa, která je použitelná pro adresování v XMPP síti, existují sice i další možnosti adresování, jako XMPP-URI, ale ty jsou určeny pouze pro použití mimo kontext XMPP sítě, např. při připojování k JID z webové stránky. Výsledná JID tedy vypadá následovně:

```
JID: [ localpart "@" ] domainpart [ "/" resourcepart ]
```

Např.: kamera161681@jablocom.com/vlakno_kamery nebo jablocom.com. [4]

2.2.3 Princip navazování relace

XMPP ke komunikaci využívá protokol TCP. První akcí klienta či serveru, který chce komunikovat s jinou entitou v síti musí být proto vytvoření TCP spojení. Následně se k vyměňování informací používají takzvané XML streamy a XML stanza. XML stream je jednosměrný kontejner obsahující všechny elementy proudící mezi dvěma entitami ze strany iniciátora („initial stream“) a pokud chce druhá entita zasílat odpovědi, musí vytvořit vlastní stream (tj. „response stream“). K otevírání či zavírání streamu se používá XML element <stream/>. XML stanza jsou základní XMPP elementy, které jsou z pohledu XML streamu na první úrovni a náleží do jmeného prostoru „jabber:client“ nebo „jabber:server“, tedy jmenovitě <message/>, <presence/> a <iq/>. Zjednodušený příklad komunikace je naznačen v tabulce číslo 1. [5]

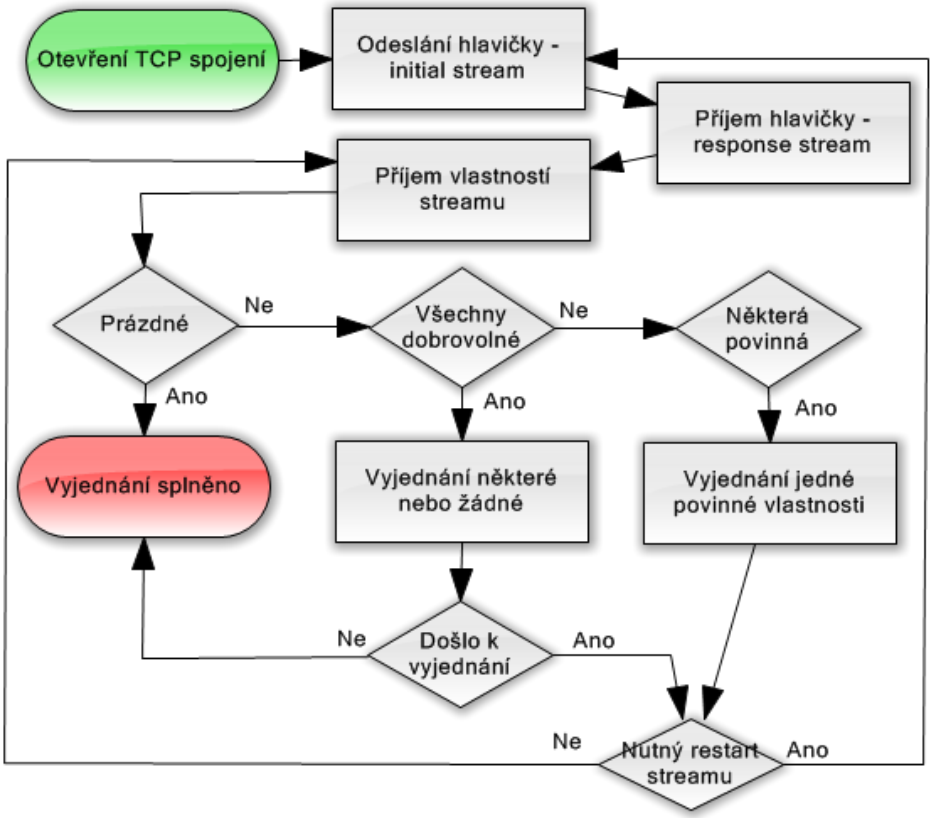
Initial stream	Response stream
<stream>	
	<stream>
<presence/>	
<message/>	
<iq type='get'/>	
	<iq type='result'/>
...	
	...
</stream>	
	</stream>

Tabulka 1: Ukázka XMPP komunikace

Entita, která vytvořený stream přijímá, funguje jako správce domény, služeb na serveru apod. Proto může požadovat splnění některých vlastností (feature) dříve, než umožní vstupující entitě přístup. Například pokud vytvoří stream XMPP klient, musí být před odesíláním požadavků na server patřičně autentizován. Je také nutné mít na paměti, že vlastnosti se mohou vztahovat k různým vrstvám (TCP, TLS, SASL,

XMPP ...) a proto je splnění vlastností několikaúrovňový proces, u kterého mohou být některé vlastnosti přidány po splnění vlastnosti předchozí a u kterého jsou některé vlastnosti povinné a jiné pouze dobrovolné. Diagram otevření XML streamu je zobrazen na obrázku č. 2. [5]

Standard popisující XMPP protokol některé základní vlastnosti definuje. Patří mezi ně například STARTTLS (zabraňuje odposlouchávání a manipulaci se streamem využitím šifrovací vrstvy TLS), SASL (metoda pro autentizování streamu vzhledem k specifickému XMPP profilu pomocí SASL), Resource Binding (poté co je klient autentizován, musí být streamu přiřazena resourcepart, tak aby server mohl přijímat XML stanza paralelně od dvou či více XMPP klientů). [5]



Obrázek 2: Diagram otevírání XMPP relace

2.2.4 Správa kontaktů v XMPP

V XMPP může uživatelův kontakt list (nebo několik kontakt listů) obsahovat neomezený počet kontaktů. Tento list se udržuje jako součást profilu uživatele, který je uložen nejčastěji přímo na domácím XMPP serveru (server, pomocí kterého klient vstupuje do XMPP sítě, určený pomocí domainpart klientova JID). Tento fakt uživateli umožňuje přidávat/upravovat kontakty na server a ten je povinen si tyto změny

ukládat, pokud možno v nemodifikované podobě. Server pak na vyžádání poskytne tento list uživateli a ten je tak schopen připojit se z jakéhokoli zařízení (notebooku, telefonu ...) a vždy mít aktualizovaný kontakt list. Zde je důležité připomenout, že uživatelův list obsahuje důvěrná data a proto přidávat, upravovat či mazat kontakty smí pouze oprávněný uživatel (typicky pouze majitel účtu). [6]

Manipulace s kontakt listem se provádí pomocí elementu <iq/> respektive jeho potomka <query/>, který je definovaný ve jmenném prostoru „jabber:iq:roster". Součástí elementu <query/> je pak žádný, jeden nebo více elementů <item/>, kde každý z nich obsahuje informaci o jednom kontaktu uživatele. Pro manipulaci s kontakt listem se pak používají čtyři specifické příkazy roster get, result, set a push. [6]

Element <item/> může obsahovat několik různých atributů a elementů. Atributy approved a ask slouží k informování o stavu či potvrzení tzv. presence subscription (viz 2.2.6), atribut jid představuje identifikátor daného kontaktu/itemu a nepovinný atribut name představuje nástroj pro fyzického uživatele, díky kterému si může daný kontakt „přejmenovat" tak, aby věděl o koho se jedná. Element <group/> se může uvnitř <item/> vyskytovat vícekrát v závislosti na tom, v kolika skupinách je uživatel umístěn. Skupiny pak primárně slouží k rozlišení různých druhů kontaktů z pohledu uživatele. [6]

Posledním atributem, který může element <item/> obsahovat, je atribut subscription. Tento atribut je velmi důležitý, určuje aktuální stav tzv. presence subscription, tedy kterým směrem mohou proudit informace o změně stavu a kterým nikoli. Může nabývat čtyř různých hodnot. První je „none", tato hodnota je také výchozí (nemusí tedy být vyplněna) a v tom okamžiku neproudí informace o stavu žádným směrem. Jestliže je hodnoty „to", tak informace o stavu proudí pouze směrem k uživateli, kontakt tak stále uživatele vidí offline. Hodnota „from" představuje opačný směr, kontakt vidí uživatelův stav a ten vidí svůj kontakt offline. Při hodnotě „both" pak stavová informace proudí oběma směry. Poslední speciální hodnotou je „remove", ta se vyskytuje pouze v příkazech typu set (odebrání kontaktu z listu) a push (informace ostatním resource o odebrání kontaktu). [6]

Chce-li uživatel získat obsah svého kontakt listu, například protože se právě přihlásil na svůj účet z dalšího zařízení, měl by ještě před odesláním initial presence (viz 2.2.5) odeslat na server příkaz roster get obsahující pouze prázdný element <query/>: [6]

```
<iq from='kamera568@jablocom.com/vlakno' id='bv1bs71f' type='get'>
 <query xmlns='jabber:iq:roster'/>
</iq>
```

Tím se z něj stává také tzv. „interested resource“, což znamená, že ho bude server průběžně informovat o změnách v jeho listu, pomocí příkazů roster push. Server mu odpoví pomocí příkazu roster result, jehož součástí je aktuální obsah kontakt listu klienta nebo pomocí elementu <iq/> obsahujícího element <error/>, pokud z nějakého důvodu není schopen sestavit odpověď. Element <query/> příkazu roster result obsahuje příslušný počet elementů <item/>. Tento počet může být roven i nule v okamžiku, kdy klient nemá ve svém listu žádný kontakt. Odpověď roster result vypadá následovně: [6]

```
<iq id='bv1bs71f' to='kamera568@jablocom.com/vlakno' type='result'>
 <query xmlns='jabber:iq:roster' ver='ver11'>
  <item jid='klient3@example.net' name='Klient3' subscription='both'>
   <group>Jablocom customers</group>
  </item>
  <item jid='klient8@example.com' name='Klient8' subscription='to'/>
  <item jid='klient4@example.cz' name='Klient4' subscription='both'/>
 </query>
</iq>
```

Přidávání, upravování a mazání kontaktů, pak uživatel provádí odesíláním příkazů roster set, jehož element <query/> obsahuje právě jeden <item/>, který je modifikován. Při přidání a úpravě kontaktu jsou součástí příkazu vybrané hodnoty atributů, dle kterých je kontakt modifikován a v případě odebrání je vložen pouze atribut subscription s hodnotou remove, na což server zareaguje odesláním příkazů presence unsubscribe, unsubscribed nebo obou (viz 2.2.6). Server je pak také povinen odeslat odpověď, a to v podobě elementu <iq/> typu result nebo error, pokud je některý z parametrů v nepořádku. Ukázka příkazu na odebrání kontaktu: [6]

```
<iq from='kamera568@jablocom.com/vlakno' id='cevre484vre' type='set'>
 <query xmlns='jabber:iq:roster'>
  <item jid='klient14@example.cz' subscription='remove'/>
 </query>
</iq>
```

Posledním druhem příkazu je push. Jeho formát je stejný jako formát příkazu set, jediný rozdíl je ve směru odeslání. Příkaz push informuje uživatele o změně některého jeho kontaktu. Význam toho příkazu spočívá v tom, že pokud je uživatel připojen v jednu chvíli z více zařízení a v jednom z nich provede změnu, bude tato změna odeslána všem ostatním zařízením (rozlišeným podle resourcepart adresy JID). [6]

2.2.5 Přenos stavové informace v XMPP

Získání stavové informace je zajištěno pomocí elementu <presence/>, plný tvar vypadá například takto: [6]

```
<presence from='kamera568@jablocom.com/vlakno' type='unavailable'
          to='klient14@example.cz/mobil'>
 <show>dnd</show> <status>Sleep</status>
 <priority>1</priority>
</presence>
```

Atributy to a from určují odesilatele a příjemce, atribut type určuje druh elementu. Vložené elementy pak specifikují stav - show určuje vybraný XMPP stav (away, chat, dnd, xa nebo NONE), status stav obecně a priority pak stanovuje prioritu daného resource. [6]

Pro přenos stavové informace existuje několik mechanismů. Prvním je takzvaný „initial presence“, který slouží k oznámení o tom, že se přihlásil nový resource uživatele – jeho XMPP klient po provedení základních přihlašovacích mechanismů odešle element presence bez atributů to a type. Ten je pak servery uživatele i kontaktů odeslán všem dostupným kontaktům a tím jim oznámí aktuální stav nově připojeného resource. Opakem tohoto mechanismu je „unavailable presence“, kde element neobsahuje atribut to, ale obsahuje type unavailable – tento mechanismus se využívá k informování všech kontaktů o tom, že se resource uživatele odpojuje a již nebude dostupný a může obsahovat element status s vysvětlením proč se uživatel odpojuje. [6]

Další je „presence probe“, tedy zaslání elementu presence s typem probe a definovanými atributy to a from. Tento mechanismus je klienty využíván jen minimálně – častěji jej zasílá server, který tak zjišťuje aktuální stavovou informaci daného kontaktu. Například proto mohl tuto informaci odeslat nově přihlášenému resource uživatele. Posledním je „presence broadcast“. Tento mechanismus je obdobou „initial presence“, tedy odeslání elementu presence bez atributů to a type. Slouží k tomu, aby uživatelův

XMPP klient mohl kdykoli během relace změnit stav uživatele. Tento element je následně prostřednictvím serverů uživatele i klientů odeslán všem dostupným resource všech klientů i všem ostatním resource uživatele. Opakem je pak „directed presence“, ten je využíván nejčastěji ke sdělení aktuálního stavu kontaktu, který nemá stavovou informaci povolenu (například protože uživatel a klient aktivně komunikují). [6]

2.2.6 Správa povolení stavové informace v XMPP

Aby bylo ochráněno soukromí uživatele protokolu XMPP, je stavová informace zasílána pouze uživatelům, kteří byli předem schváleni (kontaktům). K tomu slouží takzvané subscription oprávnění. Doba trvanlivosti těchto oprávnění je nekonečná a není tedy omezena pouze na jednu relaci jako např. v protokolu SIP. Z toho plyne, že pro odebrání tohoto oprávnění je nutné, aby uživatel manuálně vzal své povolení zpět. Správa těchto oprávnění probíhá stejně jako zasílání stavových informací pomocí elementu <presence/>, a to konkrétně typu „subscribe“, „unsubscribe“, „subscribed“ a „unsubscribed“. [6]

Žádost o získání stavové informace se zasílá pomocí elementu <presence/> typu subscribe:

```
<presence id='creg51re1' to='klient14@example.cz' type='subscribe'/>.
```

Atribut to musí obsahovat bare JID, tedy JID složené pouze z localpart a domainpart z důvodu zjednodušení celé žádosti – uživatel žádá o povolení stavové informace pro všechny komunikátory kontaktu (tzn. pro všechny resourcepart). Server uživatele pak provede kontrolu JID (např. odebrání resourcepart z JID), a pokud se nejedná o validní JID, bude vrácena chyba typu <jid-malformed/>. Následně je do požadavku přidán atribut from s hodnotou bare JID uživatele a ten je odeslán serveru kontaktu. Po odeslání tohoto požadavku dochází k odeslání příkazu roster push (viz 2.2.4, hodnota atributu subscribtion je none – zatím není stavová informace schválena v žádném směru a hodnota atributu ask je subscribe – žádost byla odeslána) všem resourcepart uživatele. Odeslání požadavku subscribe by měl server opakovat do doby, než přijde schválení/odmítnutí (četnost záleží na zvolené strategii). [6]

Server kontaktu žádost buď automaticky schválí (např. pokud již uživatel povolení získal v minulosti, nebo má kontakt nastaveno automatické schvalování pro všechny), uloží a odešle ji kontaktu později (pokud jsou všechny resource kontaktu offline) nebo ji poskytne ke schválení kontaktu. Kontakt pak na tento požadavek může reagovat schválením:

```
<presence id='1h6rv' to='kamera568@jablocom.com' type='subscribed'/>.
```

Tedy odesláním elementu presence typu subscribed. Po schválení musí server kontaktu upravit položku subscribtion v záznamu uživatele na from (případně both, pokud již došlo ke schválení z druhé strany) a informovat o tom všechny interested resource kontaktu. Dále informace putuje k serveru uživatele společně s informací o aktuálním stavu kontaktu, ten upraví záznam kontaktu (subscribtion na to/both) a odešle jak informaci o změně tak aktuální stav kontaktu všem interested resource uživatele. [6]

Nebo kontakt požadavek zamítne odesláním presence typu unsubscribed:

```
<presence id='9de' to='kamera568@jablocom.com' type='unsubscribed'/>.
```

Ten je odeslán uživatelovu serveru, který na něj reaguje odebráním hodnoty atributu ask a odesláním roster push všem interested resource. [6]

Presence unsubscribed může být odeslán i mimo kontext schvalování žádosti o stavovou informaci. V tomto případě pak zajišťuje ukončení platnosti předschválení či odebrání oprávnění na stavovou informaci. Server na tento požadavek reaguje aktualizací stavu kontaktu (z both na to, či z from na none) všem interested resource uživatele (roster push) a odesláním presence unsubscribed společně s presence unavailable serveru kontaktu:

```
<presence from='kamera568@jablocom.com' id='e6' type='unavailable'/>.
```

Server kontaktu zareaguje postupným přeposláním presence unsubscribed, roster push (změna směru stavové informace) a presence unavailable všem dostupným interested resource kontaktu (pokud není žádný resource kontaktu dostupný, měl by žádost uložit a o odebrání oprávnění kontakt informovat při prvním přihlášení). [6]

Posledním typem presence elementu je unsubscribe:

```
<presence id='e6rn4te' to='klient14@example.cz' type='unsubscribe'/>.
```

Ten uživatel odesílá v případě, že již nadále nechce dostávat informaci o aktuálním stavu kontaktu. Jeho server zareaguje zasláním roster push všem interested resource (změna subscribtion z both na from, či z to na none) a přeposláním žádosti serveru kontaktu. Server kontaktu pak žádost přepošle všem interested resource kontaktu společně s roster push (změna z both na from, či z to na none) a opačným směrem zašle presence unavailable za každý online resource kontaktu. [6]

2.2.7 Přenos zpráv v XMPP

Po autentizaci a provedení dalších nezbytných mechanismů je uživateli povoleno odesílání či přijímání XMPP zpráv, probíhající prostřednictvím elementu <message/>: [6]

```
<message from='kamera568@example.com/vlakno' id='cebc61vc'
         to='klient14@example.cz' type='headline'>
 <subject>Poplach</subject>
 <body>Ve strezenem objektu doslo k˜poplachu</body>
</message>
```

Atribut from identifikuje odesilatele zprávy a měl by obsahovat full JID, tedy včetně resourcepart. Atribut to identifikuje příjemce zprávy a nejčastěji obsahuje pouze bare JID (adresa bez resourcepart), tak aby nebylo nutné předem rozhodnout o tom, kam bude zpráva zaslána (pokud si uživatel s kontaktem vyměňují větší množství zpráv, dochází k cílení a tedy i to obsahuje full JID). Atribut type určuje typ zprávy – chat (zpráva je součástí komunikace dvou klientů), error (informace o chybě při odeslání zprávy), groupchat (komunikace více uživatelů), normal (osamocená zpráva, u které se očekává odpověď) či headline (upozornění, hlášení – osamocená zpráva, bez očekávání odpovědi). Element body obsahuje sdělení, které zpráva obsahuje. Je možné, aby zpráva obsahovala více elementů body, ty jsou pak označeny „xml:lang“ a obsahují lokalizaci zprávy do více jazyků. Element Subject obsahuje upřesňující informaci o tématu zprávy. [6]

3 Specifikace zadání

Navržení a vytvoření služby, která bude přenášet uživatelské zprávy z bezpečnostní kamery EYE-02 na IM klienta uživatele a také bude umožňovat pomocí tohoto klienta kameru ovládat a zobrazovat její aktuální stav, je velmi obecné zadání problému. Při pohledu na základní a zejména rozšiřující funkcionalitu protokolu XMPP je patrné, že lze službu rozšiřovat o celou řadu mechanismů poskytujících různé služby. Dále je také třeba mít na zřeteli, že společnost Jablocom s. r. o. má existující infrastrukturu pro práci s daty, která jsou zasílána směrem od kamery ke klientovi a také směrem od klienta ke kameře a nebylo by tedy vhodné navrhnout službu v rozporu s těmito postupy. Proto bylo po konzultaci se zadavatelem projektu navrženo následující upřesnění zadání:

1. Realizace databáze, která bude prostředníkem mezi infrastrukturou společnosti Jablocom s. r. o. a vytvořenou službou. Do této databáze by mělo být možné uložit všechny příchozí události: zpráva od klienta, zpráva od kamery, změna stavu kamery, přidání adresy klienta do kontakt listu kamery, odeslání souboru z účtu kamery, žádost klienta o videohovor, ... a další doplňující informace jako je stav zpracování události a také podrobnosti zpracování.
2. Navržení a implementace WCF služby zajišťující zápis událostí přicházejících ze strany společnosti do databáze a jejíž prostřednictvím bude možné získávat informace o událostech ze strany klienta a ty filtrovat v závislosti na různých parametrech. Pomocí služby by také mělo být umožněno sledovat stav zadané události a reagovat tak například na její nedoručení nebo získat seznam všech aktuálně vytvořených účtů kamer a jejich stav (běžící/zastavený).
3. Výběr domácího XMPP serveru, přes který bude služba komunikovat s vnějším světem, v závislosti na potřebné funkcionalitě a požadavcích společnosti Jablocom.
4. Zprovoznění a nastavení XMPP serveru na testovacím serveru společnosti Jablocom tak, aby bylo umožněno testování vytvořené služby a zároveň byla zaručena maximální bezpečnost přenášených dat i serveru samotného.
5. Prozkoumat možnosti protokolu XMPP směrem k ověření přenášených dat – tedy ověření, zda odeslaná data skutečně dorazila na IM klienta uživatele.
6. Implementace vlákna kamery, které bude v periodických intervalech přebírat jemu určené události z databáze a na základě těchto záznamů odesílat určené

XML stanza serveru (odeslání zprávy klientovi, změna stavu, přidání/odebrání adresy klienta z kontakt listu nebo navazání videohovoru), či ve speciálních případech komunikovat přímo s klientem (odesílání souboru). Vlákno kamery také musí asynchronně reagovat na příchozí XML stanza – příchozí zpráva, žádost o videohovor, potvrzení žádosti presence subscribtion aj. a v závislosti na nich zapisovat události do databáze.

7. Prověřit možnosti uskutečňování videohovoru prostřednictvím protokolu XMPP se zaměřením na XMPP klienty společnosti Google Inc. – Google talk či Google chat.

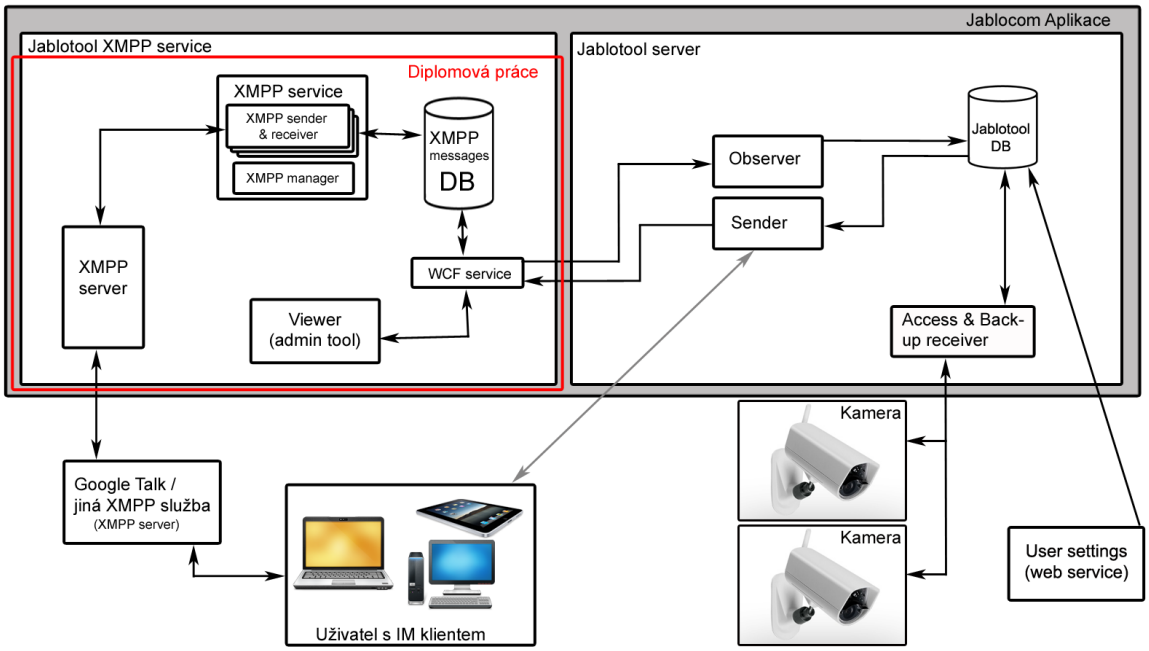
8. Návrh a implementace hlavní služby, která se bude starat o spouštění a ukončování vláken kamer v závislosti na tom zda má vlákno ve svém kontakt listu adresu některého z klientů společnosti Jablocom s. r. o. Dále by služba měla v periodických intervalech kontrolovat běh spuštěných vláken a v případě, že narazí na problém, vlákno restartovat a uložit o tom zprávu do logu.

9. Vytvoření webové stránky, která bude demonstrovat funkcionalitu zmíněné WCF služby. Webová stránka bude umožňovat dvojí přihlášení: jednak jménem a heslem kamery, kde si bude možné prohlédnout pouze údaje z vybrané kamery a pak také administrátorským jménem a heslem, kde bude možné celou aplikaci spravovat – vytvářet účty kamer, zadávat jim příkazy či zobrazovat příchozí a odchozí události a filtrovat je dle zadaných kritérií.

K vytvoření všech aplikací bude použito vývojové prostředí Microsoft Visual Studio a platforma .NET. Konkrétně pro vývoj hlavní služby, vlákna kamery a WCF rozhraní bude použit programovací jazyk C# a webová stránka bude kombinací C#, JavaScript a HTML pomocí frameworku ASP.NET MVC. Jako databázový systém bude použit Microsoft SQL Server. Služba pak musí fungovat na systémech Microsoft Windows, konkrétně pak na Windows Server.

4 Realizace služby

Na základě upřesněného zadání (viz 3) je tedy celkové schéma služby následující:



Obrázek 3: Schéma výsledné aplikace

Informace z kamery (poplachy, režim kamery, obrázky z kamery a další) jsou společně s nastavením získaným od klienta (povolení služby XMPP, zadání XMPP adresy pro komunikaci s kamerou apod.) uchovávány v „Jablotool DB“ (databáze společnosti Jablocom). Vybrané informace (typicky ty, u kterých si uživatel zvolí, že je chce získávat prostřednictvím XMPP klienta), jsou pak přebírány službou, zde pojmenovanou „Sender“ – ta je prostřednictvím WCF rozhraní zapisuje do databáze vytvořené služby „XMPP messages DB“. Odtud informace/povely putují k vytvořené službě („XMPP service“), která je zpracovává a na jejich základě provádí příslušné operace jako je spuštění vlákna kamery (hlavní služba) nebo odeslání zprávy uživateli (vlákno kamery). Komunikace s klientem je pak realizována sestavením příslušného XML stanza, které je předáno domácímu XMPP serveru. Ten vytvoří spojení s XMPP serverem klienta a data mu předá. XMPP server klienta tyto informace předá přímo klientovi, který může být připojen například ze svého mobilního telefonu či notebooku.

Komunikace samozřejmě stejně může probíhat i z druhé strany, kde domovský XMPP server zasílá získaná XML stanza vláknu kamery a to reaguje jejich zpracováním – změnou svého vnitřního stavu (odebrání klienta z kontakt listu) nebo zapsáním příslušné události do „XMPP messages DB“ (příchozí zpráva, žádost o vi-

deohovor), odkud je prostřednictvím WCF rozhraní přebírá služba „Observer“ a přes „Jablotool DB“ dochází k jejich zpracování a odeslání na kameru.

4.1 Výběr XMPP serveru

V několika následujících podkapitolách dojde k představení (vybrané informace, popis instalace a možnosti konfigurace) vybraných open source XMPP serverů, které by mohly být využity pro účely spojení služby s okolním světem – Openfire (4.1.1), Ejabberd (4.1.2), Jabberd 2.x (4.1.3), Tigase (4.1.4) a Prosody (4.1.5). Díky otevřenosti technologie XMPP to samozřejmě nejsou všechny možnosti a XMPP serverů existuje celá řada – oficiální seznam je možné nalézt na webovém portálu technologie XMPP http://xmpp.org/xmpp-software/servers/. Ovšem tyto zmíněné servery spojuje několik faktů – jejich vývoj stále pokračuje, jsou dostupné bezplatně a jejich použití v komerčních aplikacích je možné.

V podkapitole 4.1.6 dojde k porovnání vlastností těchto serverů a také k výběru serveru, který bude použit v této aplikaci. V podkapitole 4.1.7 bude pak popsáno jak server nastavit s ohledem na plnou funkčnost a maximální bezpečnost dat přenášených vyvíjenou službou i služby samotné.

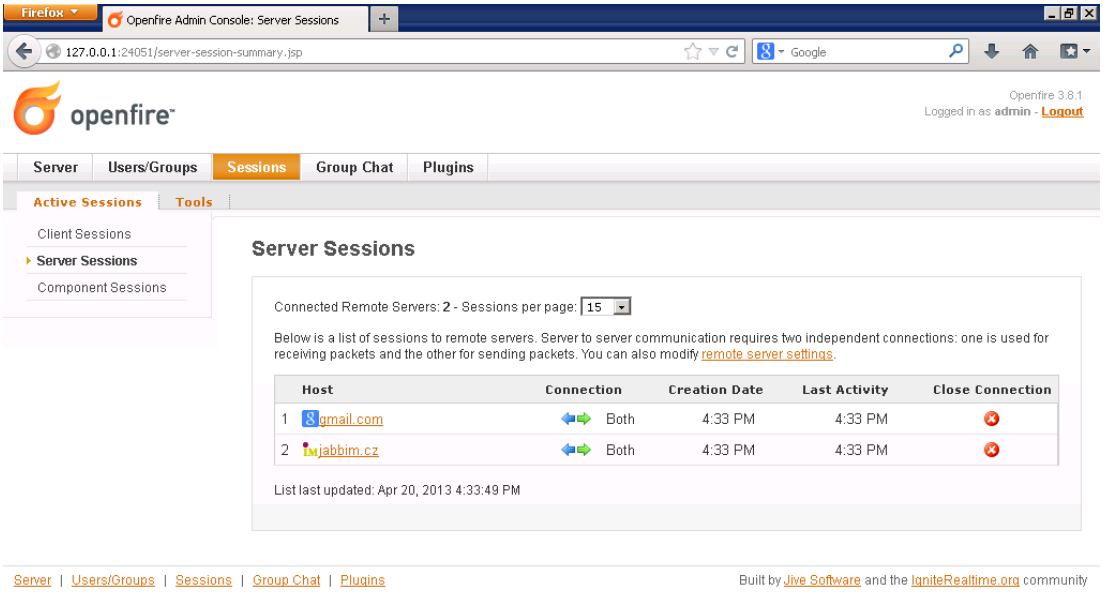
4.1.1 Openfire

Openfire je XMPP server spravovaný open source komunitou Ignite Realtime vedenou odborníky ze společnosti Jive Software. Komunita se zaměřuje na zmíněnou technologii XMPP a spravuje několik projektů, mezi které patří server Openfire, klient Spark, jeho webová verze SparkWeb či knihovny sloužící k vytvoření vlastní XMPP aplikace – Smack (Java), XIFF (Flash). [7]

Server openfire je napsaný v programovacím jazyce Java a je vydaný pod licencí Apache 2.0, což umožňuje jeho neomezené (místně i časově) použití a to i v oblasti komerční sféry. Aktuální verze serveru je 3.8.1 vydaná 3. 3. 2012. [7] Samotná instalace tohoto serveru je velmi snadná (dostupná v češtině), uživatel je dotázán pouze na souhlas s licencí serveru a na složku kam má být server nainstalován. Po instalaci se server sám automaticky spustí a po přechodu do „admin console“ při stisku „Launch Admin“ dochází k úvodní konfiguraci.

Zde si uživatel nejdříve zvolí překlad admin console (k dispozici je opět i čeština), následně vyplní doménu serveru na kterém Openfire poběží a dva porty, které jsou používány k obsluze admin console zvenčí. Dalším krokem je volba databáze, kterou

server bude používat pro uchovávání nastavení či dat o klientech – lze využít vestavěnou databázi nebo určit databázi vlastní – pak je nutné vyplnit typ databáze, jméno a heslo pro přístup a další parametry (maximální počet souběžných připojení, časový limit připojení apod.). Následuje možnost propojit server s adresářovým serverem (OpenLDAP, Active Directory apod.) či technologií Clearspace, které podporují pokročilou správu uživatelských účtů – opět nutné vyplnit autorizační údaje a další údaje jako mapování klientů na adresáře. Posledním údajem je administrátorský účet, pomocí kterého se pak lze připojit do admin console a provádět změny nastavení serveru Openfire.



Obrázek 4: Openfire – admin console – relace s XMPP servery

Admin console má pět hlavních kategorií, ve kterých lze sledovat aktuální chod serveru a provádět příslušné změny. Kategorie „Plugins“ slouží k rozšiřování funkcí serveru, kde se nachází seznam aktuálně nainstalovaných rozšíření a také seznam dostupných rozšíření – v němž jsou oficiálně schválená a podporovaná rozšíření, která jsou uživateli používána nejčastěji a samozřejmě je tu i možnost přidat rozšíření vlastní. Kategorie „Group chat“ slouží k vytváření a správě konferenčních místností na serveru, prostřednictvím kterých lze uskutečňovat například hromadné vnitrofiremní konference.

Kategorie „Sessions“ obsahuje aktuálně připojené aplikace, tedy klienty připojené na svůj účet, ale i navázané spojení s ostatními XMPP servery (viz obr. č. 4). Zde lze také odeslat hromadnou zprávu všem klientům serveru. V kategorii „Users/Groups“ je možné spravovat skupiny – přidávat/mazat aj. nebo účty klientů na serveru –

vytvářet či mazat účty, přiřazovat jim údaje (email, jméno), měnit jejich heslo, přidávat/odebírat jim kontakty z kontakt listu, blokovat je či je přidávat do skupin. Kategorie „Server“ pak slouží k nastavení vybraných vlastností serveru, či ke změně parametrů zadaných při inicializaci.

4.1.2 Ejabberd

Ejabberd je další server, který je spravován svou vlastní komunitou za podpory komerční společnosti. Jeho vývoj začal v roce 2002 programátor jménem Alexey Shchepin, díky jeho předchozímu úspěchu s XMPP klientem Tkabber a chuti pracovat s programovacím jazykem Erlang, ve kterém je Ejabberd vytvořen. V současnosti se do jeho vývoje intenzivně zapojuje společnost ProcessOne, která podniká právě v oboru instant messagingu se zaměřením na technickou podporu, školení, konzultace a vývoj software na zakázku. [8]

Jak bylo zmíněno Ejabberd je vytvořen v jazyce Erlang a je publikován pod licencí GPL v2, která se vztahuje primárně na upravování a další šíření díla, ale jeho používání a provoz není upraven a to ani v oblasti komerční sféry. Aktuální verze serveru Ejabberd je 2.1.11 vydaná 4. 5. 2012. Server je možné stáhnout v podobě zdrojových kódů nebo instalátoru pro vybraný operační systém (Windows, Linux či Mac OS). [8] Instalace serveru při použití instalátoru pro Windows je poměrně jednoduchá, i když bohužel není k dispozici česká lokalizace. Během instalace je uživatel dotázán na několik údajů – souhlas s licencí, cílová složka, doména serveru, na kterém poběží, jméno a heslo administrátorského účtu a zda bude server částí clusteru XMPP serverů (v případě že ano tak i na jeho označení). Po instalaci je možné server ihned provozovat pomocí vytvořených aplikací Start/Stop.

Konfigurace serveru může probíhat dvěma způsoby. První možností je využití grafického „admin interface“ (po spuštění serveru je spuštěná webová stránka s odkazem). Zde je pět základních nabídek. V nabídce „Seznamy přístupových práv (ACL)“ lze povolit/zakázat přístup vybraným uživatelům a těm povoleným přiřadit určitou skupinu, na základě které budou moci v budoucnu přistupovat k vybraným službám serveru. V druhé nabídce „Pravidla přístupů“ pak lze přidávat oprávnění přístupu uživatelů k vybraným službám serveru.

V nabídce „Virtuální hostitelé“ pak lze zjistit počet registrovaných a online uživatelů k dané JID doméně. Po otevření příslušné JID domény se zobrazí nabídka specifická právě pro danou doménu. Jejím prostřednictvím lze upravit výše zmíněné

údaje globálně, prohlížet a upravovat záznamy uživatelů domény (změna hesla, přidání kontaktu do kontakt listu apod.), přidávat moduly (rozšiřující funkce) jednotlivým serverům této domény plus upravovat jejich parametry a další akce specifické pro danou JID doménu (zobrazení statistik, sdílených skupin aj.).

Nabídka „Uzly“ obsahuje seznam všech serverů i vzhledem k JID doménám (Ejabberd podporuje tzv. virtual hosting, tedy na jednom serveru, může být najednou až několik XMPP domén). Po otevření nabídky lze jednotlivým strojům nastavovat způsob práce s databází (povolit/zakázat vytvoření kopie v paměti, ...), zálohovat či obnovovat data z databáze nebo data o klientech serveru, upravovat přístupové porty a IP adresy, zobrazit statistiku či daný uzel aktualizovat. V sekci „Statistiky“ jsou pak zobrazeny aktuální informace o relacích XMPP serveru.

Konfigurace některých vlastností se provádí ve specifickém formátu, ale ke každé nabídce je připravena v horním pravém rohu nápověda ([Guide: název]). Tento specifický formát vychází z formátu používaného v konfiguračním souboru. To je zároveň druhá možnost konfigurace, tzn. editovat přímo soubor ejabberd.cfg, který se nachází v instalačním adresáři – podadresář conf. Zde je důležité mít na zřeteli, že konfigurace se ukládá v databázi serveru, která se při každém startu serveru aktualizuje právě ze souboru ejabberd.cfg, ale při změně parametrů z admin interface k úpravě souboru nedochází. Tzn. oba způsoby by se neměly kombinovat a chce-li uživatel využívat ke konfiguraci grafické rozhraní, měl by soubor udržovat prázdný.

4.1.3 Jabberd 2.x

XMPP server Jabberd 2.x je často mylně označován jako další verze serveru Jabberd 1.x. Ve skutečnosti je tento server jiným projektem, který vytvořili vývojáři Jeremie Miller, Thomas Muldowney, Ryan Eatmon, Robert Norris a Tomasz Sterna, za účelem dodržování aktuálních standardů komunity XSF a zvýšení výkonu. Server je primárně určen pro instalaci na unixových systémech, ale jeho zdrojové kódy jsou psané v jazyce C++, dostupné v rámci služby GitHub a nejsou na systému závislé, což umožňuje překlad serveru i na operačním systému Windows. [9]

Server Jabberd 2.x je publikován stejně jako server Ejabberd pod licencí GPL verze 2, což znamená že je použitelný i pro komerční účely. Aktuální verze serveru je 2.2.17 vydaná v srpnu roku 2012. Vzhledem k tomu, že není k dispozici instalátor ale pouze zdrojové kódy je instalace serveru Jabberd obtížnější. Před samotným překladem je navíc třeba získat několik nezbytných knihoven tak, aby server podpo-

roval všechny zmíněné funkce např. libgsasl pro podporu technologie SASL, libexpat obsahující parser XML elementů, libeay, ssleay32 pro podporu SSL a řadu dalších. Poté stačí celý projekt sestavit pomocí Visual studia či překladače MinGW. [9]

Konfigurace serveru respektive přidání jednotlivých jeho součástí se provádí již během překladu pomocí vybraných parametrů příkazu „./configure“. Konfigurace parametrů těchto součástí či samotného serveru se poté provádí úpravou konfiguračních souborů sm.xml, c2s.xml, s2s.xml a řady dalších v adresáři /etc. Například databáze na serveru MySQL se nastaví úpravou souboru sm.xml: [9]

```
<mysql>
 <host>localhost</host> <port>6841</port>
 <dbname>JabberdDB</dbname>
 <user>jabbUser</user> <pass>secret</pass>
 <transactions/>
</mysql>
```

4.1.4 Tigase XMPP server

Počátky XMPP serveru Tigase se datují do roku 2004, kdy tento projekt odstartoval vývojář Artur Hefczyc. V současné době je server spravován firmou Tigase inc., která vznikla právě díky komerčnímu úspěchu tohoto serveru a která podniká v oboru XMPP technologií – rozšiřující moduly či komponenty serveru, vývoj specifických XMPP klientů, technická podpora pro velké projekty. Server je naprogramován v jazyce Java a jeho zdrojové kódy podléhají licenci AGPL v3. Aktuální verze serveru je 5.1.4 vydaná 14. 1. 2013. Server nabízí dvě varianty instalace – grafický (doporučený) a konzolový instalátor (přepis oken do konzole). Před započetím instalace je nutné zprovoznit na počítači JDK (minimální verze 1.6), stáhnout soubor ve formátu jar a spustit ho (java -jar cesta). [10]

Poté se zobrazí grafický průvodce instalací skládající se z 21 kroků. Nejdříve je nutné zadat cestu k JDK a zvolit, zda chce uživatel server pouze nainstalovat či rovnou i konfigurovat. Následuje seznámení se serverem a jeho licencí, volba instalačního adresáře, volba komponent k instalaci (zde jsou důležité zejména databázové ovladače) a samotná instalace serveru. Po instalaci jsou vytvořeni dva zástupci: jeden spouští server manuálně a druhý umožňuje nainstalovat službu, která bude server spouštět automaticky. V dalších krocích dochází ke konfiguraci serveru (nejdříve je zobrazena aktuální nebo výchozí konfigurace serveru).

Součástí konfigurace je zadání XMPP domény, administrátorského účtu a hesla, volba databáze (možnost zvolit oddělenou DB pro autentizaci), přidání nepovinných komponent, povolení/zakázání připojení serveru do clusteru a volba nástrojů pro logování. Dále se pak uživateli zobrazí nabídka se všemi podporovanými pluginy, ze kterých si může zvolit ty, které chce na serveru používat (SASL, TLS, Zlib komprese, offline ukládání zpráv, ...). Následuje nastavení zvoleného databázového systému (pro základní i autentizační databázi) – Tigase vytvoří všechny potřebné tabulky a vlastní účet, přes který bude s databází pracovat. Výhodou je, že po konfiguraci lze vytvořit skript, který příští konfiguraci provede automaticky.

Další konfigurace serveru se provádí úpravou souboru „etc/init.properties“ přidáním vybraných příkazů, které jsou zdokumentovány na domovské stránce serveru. Pokročilou konfiguraci je pak možné provádět několika způsoby úpravou souboru „tigase.xml“, úpravou hodnot v databázi či úpravou parametrů přímo v paměti za běhu serveru pomocí skriptů. Skripty lze vytvářet ručně v některém z podporovaných programovacích jazyků, či použít předvytvořené skripty „scripts/admin“. K zasílání skriptů je pak třeba klient, který techniku podporuje – na domovské stránce lze nalézt příklad s klientem PSI. [10]

4.1.5 Prosody

Prosody je relativně mladý XMPP server (první verze byla vydána až v prosinci 2008), jehož autorem je Matthew Wild. Server je zaměřen na jednoduchost nastavení a minimální nároky na hardware, z pohledu vývojáře pak na jednoduchost přidávání nové funkcionality. Server je naprogramován v jazyce Lua (odlehčený procedurální jazyk navržený jako skriptovací, používaný často k úpravě chování programu bez zasahování do jeho jádra – např. úprava GUI ve hře World of Warcraft) a vydaný pod licencí MIT/X11. Server je aktuálně ve verzi 0.8.2, která byla vydána 20. 6. 2011. [11]

Instalace serveru je velmi snadná – stačí pouze souhlasit s licencí, zadat instalační adresář a server je nainstalován. Po instalaci jsou k dispozici tři zástupci: odkaz na webovou stránku, zástupce pro spuštění serveru a odkaz na konfigurační soubor. Konfigurační soubor se skládá ze tří částí. V první části se nachází globální nastavení společné pro všechny domény spravované serverem. Další část začíná klíčovým slovem „VirtualHost“ následovaným názvem domény v uvozovkách. Veškeré nastavení za touto částí se vztahuje pouze k uvedené doméně, domén může být uvedeno i více

a pro každou různé nastavení. Poslední částí je pak přidání komponent serveru, kde může být přidán například víceuživatelský chat či proxy pro přenos souboru. Ukázka nastavení:

```
admins = { "admin1@jablocom.com", "admin2@jablocom.cz" }
modules_enabled = { "roster"; "saslauth"; };
storage = "sql"
 sql = { driver = "MySQL"; database = "xmpp"; host = "pokus.jabl.cz";
         username = "admin"; password = "heslo"; }
VirtualHost "jablocom.cz"
 modules_enabled = { "register"; } allow_registration = true;
VirtualHost "jablocom.com"
 disallow_s2s = true;
Component "conference.jabl.cz" "muc"
 restrict_room_creation = admin;
```

V ukázce došlo k nastaveni dvou administrátorů, povolení modulů pro získání kontaktů a SASL. Databáze je zajištěna pomocí databázového systému MySQL. Doména jablocom.cz umožňuje registraci uživatelů ze svého klienta a doména jablocom.com nedovoluje spojení se vzdáleným serverem tzn. komunikace je uzavřena pouze v rámci této domény. Na adrese conference.jabl.cz je pak konferenční služba, ve které smí zakládat místnosti pouze administrátoři.

4.1.6 Shrnutí a výběr

V předchozích kapitolách byly představeny jednotlivé XMPP servery a také možnosti jejich konfigurace a ovládání. V následujících několika tabulkách budou srovnány jejich základní vlastnosti a možnosti nastavení, které jsou důležité z hlediska vytváření výsledné služby. V tabulce číslo 2, jsou uvedeny obecné informace tedy licence, programovací jazyk, databázové systémy, které jsou podporovány k ukládání dat serveru, rozšiřitelnost pomocí pluginů apod. V tabulce číslo 3 je pak zhodnocena možnost nastavení vybraných služeb serveru a v tabulce číslo 4 je zhodnocení podpory vybraných XMPP standardů.

Nevýhodou serveru Prosody je, že jedinou možností jak vytvářet uživatele na serveru je takzvaná in-band registrace, tedy registrace z XMPP klienta. Tím by bylo umožněno vytvořit si účet komukoli, tedy i potenciálnímu útočníkovi – nástroj prosodyctl není pod Windows aktuálně dostupný. Další nevýhodou je absence podpory

MS SQL, který je používán společností Jablocom s. r. o. Výhodou i nevýhodou je pak jeho snadná konfigurace, která budí dojem, že je server určen spíše pro komunikaci v malé vnitrofiremní síti, než pro větší projekt, kterým tato služba z pohledu XMPP serveru bude.

Název serveru	**Openfire**	**Ejabberd**	**Jabberd 2.x**	**Tigase**	**Prosody**
Domovská stránka	[7]	[8]	[9]	[10]	[11]
Programovací jazyk	Java	Erlang	C/C++	Java	Lua
Výchozí databáze	HSQLDB	Mnesia	Filesystem	Derby DB	Filesystem
Podporované databáze	MySQL, Oracle, MS SQL, PostgreSQL, IBM DB2	MySQL, MS SQL, PostgreSQL, **ODBC**	MySQL, Oracle, PostgreSQL, SQLite, Berkeley DB	MySQL, MS SQL, PostgreSQL	MySQL, PostgreSQL, SQLite, MongoDB
Možnost rozšířit moduly/pluginy	Ano	Ano	Ne	Ano	Ano
Tutorialy pro tvorbu pluginů	Ano	Ano	Ne	Ano	Ano
Dostupná dokumentace API	Ano	Ne	Ne	Ano	Ne
Licence	Apache v2.0	GPL v2	GPL v2	AGPL v3	MIT/X11
Grafické rozhraní pro správu	Ano	Ano	Ne	Ne	Ne

Tabulka 2: Srovnání XMPP serverů – Obecné informace

Server Jabberd 2.x je určen spíše pro unixové systémy, jeho instalace i konfigurace pod systémem Windows je tudíž poměrně náročná. Další nevýhodou je absence podpory databázového systému MS SQL, absence podpory nového XMPP standardu vydaného v roce 2011 [5] a nemožnost rozšířit funkce serveru přidáváním modulů/pluginů.

Název serveru	**Openfire**	**Ejabberd**	**Jabberd 2.x**	**Tigase**	**Prosody**
SASL	Ano	Ano	Ano	Ano	Ano
TLS	Ano	Ano	Ano	Ano	Ano
Zákaz registrace uživatelů z klienta	Ano	Ano	Ano	Ano	Ano
Možnost alternativní registrace	Plugin	XML-RPC	Externí DB	Skript	prosodyctl
Zákaz anonymního přihlášení	Ano	Ano	Ano	Ano	Ano
Zákaz změny hesla z klienta	Ano	Ano	Ano	Ne	Ano
Zákaz subscribe dotazů	Plugin	Ano	Ne	Ne	Ne
Virtual Hosts	Ne	Ano	Ano	Ano	Ano
Připojení do clusteru	Plugin	Ano	Ne	Ano	Ne
Ukládání offline zpráv	Ano	Ano	Ano	Ano	Ano
File transfer proxy	Ano	Modul	Externě	Externě	Ano
Stun server	Plugin	Ano	Ne	Externě	Modul

Tabulka 3: Srovnání XMPP serverů – Možnosti nastavení

Zbývající servery mají z pohledu projektu všechny důležité součásti a o výběru tak rozhodla přívětivost obsluhy a nastavení a také snadnost rozšíření o vybrané moduly (například absence STUN serveru u Tigase – dostupný pouze v komerční variantě). Proto byl pro projekt vybrán server Openfire, jehož grafické rozhraní je velmi jednoduché a intuitivní, tudíž nebude třeba pro jeho obsluhu dlouhé zaškolování jako je to v případě serveru Tigase, který se ovládá pomocí skriptování z XMPP klienta či úpravou inicializačních souborů nebo Ejabberd, který má také grafické rozhraní, ale úprava funkce modulů je odvozena z konfiguračního souboru, díky čemuž je nutné před úpravami pročíst manuál, jak danou funkcionalitu upravit.

Název serveru	**Openfire**	**Ejabberd**	**Jabberd 2.x**	**Tigase**	**Prosody**
RFC-3920 XMPP: Core	Ano	Ano	Ano	Ano	Ano
RFC-3921 XMPP: IM and Presence	Ano	Ano	Ano	Ano	Ano
RFC-6120 XMPP: Core	Ano	Ano	Ne	Ano	Ano
RFC-6121 XMPP: IM and Presence	Ano	Ano	Ne	Ano	Ano
XEP-0079 Advanced Message Processing	Ne	Částečně	Částečně	Ano	Ne
XEP-0022 Message Events	Ne	Ano	Ano	Ne	Ne
XEP-0184 Message Receipts	Ano	Ano	Ano	Ano	Ano
XEP-0065 SOCKS5 Bytestreams	Ano	Ano	Externě	Ano	Ano
XEP-0167 Jingle RTP Sessions	Ano	Ano	Ano	Ano	Ano
XEP-0096 File Transfer	Ano	Ano	Ano	Ano	Ano
XEP-0115 Entity Capabilities	Ano	Ano	Ne	Ne	Ne

Tabulka 4: Srovnání XMPP serverů – Podpora standardů

4.1.7 Nastavení

V této kapitole bude popsáno nastavení serveru Openfire takovým způsobem, aby byla zaručena plná funkce služby s ohledem na bezpečnost její i serveru. Pro změnu nastavení je nutné, aby se administrátor přihlásil do admin console (viz obr. 4) pomocí účtu zadaného během instalace. V nabídce *Server – Server Manager – Server Information* je možné provést úpravu portů, přes které se bude se serverem pracovat – porty pro XMPP komunikaci musí zůstat ve výchozí hodnotě, důležité je pak nastavení portů „Admin Console Port“, jelikož ty jsou využívány hlavní službou či administrační stránkou.

Nastavení v nabídce *Server – Server Settings* pak lze rozdělit do několika skupin dle důležitosti. Pro správný chod služby je důležité nastavit v nabídce *Client Connections* položku pro dobu odhlášení nečinného klienta přibližně na 100 sekund (vlákno kamery se pravidelně hlásí v intervalu 90 sekun + rezerva) a v případě

neohlášení odeslat XMPP ping request, na který vlákno umí odpovědět tak, aby nebylo odhlášeno. V nabídce *Server to Server* musí být povolena interakce se vzdálenými servery, aby mohlo dojít ke komunikaci se serverem klienta. V nabídce *File Transfer Settings* by měl být povolen proxy server pro přenos souborů mezi klienty, pokud je ovšem server dostupný pod veřejnou IP adresou není povolení proxy server nezbytně nutné, vlákno kamery dokáže odesílat soubory i přímo (více viz kapitola číslo 4.3.7). Pokud je ovšem proxy server povolen, je nutné nastavit mu další dva parametry (nabídka *Server – Server Manager – System Properties*): „xmpp.proxy.externalip“ na hodnotu odpovídající IP adrese přes kterou je server veřejně dostupný a „xmpp.proxy.transfer.required“ na hodnotu „false“. Adresa proxy serveru, uvnitř serveru Openfire, je „proxy.doména“.

Kvůli bezpečnosti je doporučeno zakázat v nabídce *Registration & Login* všechny položky (tedy inband registrace, změna hesla i anonymní přihlášení) dále zde může být vyplněna IP adresa počítače, na kterém poběží služba tak, aby se k serveru bylo možné přihlásit jen odtud. V nabídce *Security Settings* by pak mělo být vyžadováno zabezpečené připojení jak pro komunikaci s klientem, tak zejména se vzdálenými servery (zde pozor na fakt, že ne všechny XMPP servery přijímají self-signed certifikáty, tudíž by se mohlo stát, že ke spojení se serverem po zabezpečené lince nedojde). Externí komponenty (nabídka *External Component*), manažeři připojení (*Connection Managers*), přihlašování přes HTTP (*HTTP binding*), logování zpráv (*Message Audit Policy*), ukládání klientských dat (*Private Data Storage*), proxy server pro média (*Server – Media Services*) nejsou k funkci služby potřebné, tudíž mohou být zakázány.

Další doporučení je nastavit řešení konfliktů (*Resource Policy*) na hodnotu Always Kick, tak aby nově příchozí spojení mělo vždy přednost před stávajícím (neočekávaná chyba, přihlášení útočníka apod.). Offline zprávy by měly být uchovávány tak aby na ně vlákno kamery mohlo reagovat například až po restartu, tzn. v nabídce *Offline Messages* nastavit Always Store nebo Store or Bounce (v tomto případě je nutné stanovit maximální limit offline zpráv). Metoda komprese (*Compression Settings*) by měla být povolena, kvůli menšímu vytížení spojení zejména se vzdálenými servery.

Mimo zmíněného nastavení je na serveru dále nutné vytvořit skupinu s vybraným názvem (např. robots/cameras), do které budou automaticky zařazovány účty

kamer a také na jejím základě budou vyhledávány (*Users/Groups – Groups – Create New Group*). Skupinový chat není pro funkci služby nutný, lze tedy odstranit všechny služby či je nastavit tak, aby nebylo možné zakládat místnosti (*Group Chat – Group Chat Settings*).

Poslední položkou jsou rozšíření serveru (*Plugins*). Předinstalovaný plugin „Search“ není pro chod služby třeba, lze ho tak odinstalovat či zakázat (*Server – Server Settings – Search Service Properties*). Pro chod vytvořených aplikací je pak nutné nainstalovat (*Available Plugins*) dvě rozšíření. Prvním je „User Service“, to umožní přidávání účtů pomocí HTTP příkazů, což je potřebné proto, aby hlavní služba mohla přidávat účty kamer – účty kamer není možné přidávat přímo do databáze, jelikož server Openfire podporuje cachovaní tzn., že během chodu do databáze zapisuje změněné hodnoty ale ke čtení hodnot dochází pouze při startu serveru. Toto rozšíření je také třeba povolit a zvolit bezpečné heslo (*Server – Server Settings – User Service*).

Druhým rozšířením je „Presence Service“, to slouží k získání aktuálního stavu vybraného účtu na serveru (ve formě ikony, textu nebo XML příkazu), což je nutné pro zobrazení aktuálního stavu kamery v administraci. Pro správnou funkci je potřebné, aby byl stav kamery zobrazen komukoliv (*Server – Server Settings – Presence Service*, hodnota Anyone). Server Openfire podporuje ještě řadu dalších rozšíření, která lze využít k zabezpečení či podpoře služby, např: „Packet Filter“ (umožňuje filtrovat/logovat příchozí XML stanza od vybraných uživatelů apod.), „Content Filter“ (sledování obsahu zpráv a upozorňování administrátora), „Jingle Nodes Plugin“, „STUN server plugin“ (podpora videohovorů) apod.

4.2 Databáze

Databáze systému (výsledné schéma viz obr. 5) musí pojmout několik typů požadavků s tím, že každý ze zmíněných požadavků má různý počet údajů, které jsou nutné k jeho obsloužení. Tabulka číslo 5 obsahuje seznam různých typů požadavků, jejich popis a povinné údaje. Na základě těchto dat by bylo na místě vytvořit tabulku se společnými údaji a zbytek údajů rozdělit do 11 tabulek. Ovšem je třeba myslet na to, že nejčastějším dotazem na výslednou databázi bude právě požadavek o získání všech neobsloužených požadavků patřících pod vybranou kameru, to by znamenalo časté skládání všech 11 tabulek. Proto bylo po konzultaci se zadavatelem projektu rozhodnuto, že bude lepší variantou ignorovat velké množství neobsazených polí a všechny tyto položky vložit do dvou tabulek, které budou požadavky rozdělovat na základě

toho, zda přišly ze strany kamery či ze strany klienta a spojujícím prvkem bude právě tabulka vysvětlující typ požadavku.

ID	Označení	Popis	Povinné údaje
1	MsgOut	Zpráva od kamery	Jméno kamery, adresa uživatele, subjekt zprávy, obsah zprávy
2	Status	Změna stavu kamery	Jméno kamery, XMPP status, status
3	MsgIn	Zpráva od klienta	Jméno kamery, adresa uživatele, subjekt zprávy, obsah zprávy
4	ContAdd	Přidání klienta do kontakt listu kamery	Jméno kamery, adresa uživatele, uvítací zpráva
5	ContDel	Odebrání klienta z kontakt listu kamery	Jméno kamery, adresa uživatele
6	RobotAdd	Spuštění vlákna pro obsluhu požadavků	Jméno kamery, heslo
7	RobotDel	Zastavení vlákna pro obsluhu požadavků	Jméno kamery
8	Video	Videohovor od kamery	Jméno kamery, adresa uživatele, informace o audio a videostreamech (IP adresa, port, seznam kodeků, protokoly, typ streamu, jméno a heslo pro stream)
9	FileMsg	Zpráva od kamery obsahující soubory	Jméno kamery, adresa uživatele, subjekt zprávy, obsah zprávy, seznam souborů
10	VideoClient	Videohovor od klienta	Jméno kamery, adresa uživatele, informace o audio a videostreamech (IP adresa, port, seznam kodeků, protokoly, typ streamu, jméno a heslo pro stream)
11	RequestVideoClient	Požadavek o videohovor od klienta	Jméno kamery, adresa uživatele

Tabulka 5: Vyvíjená aplikace – Druhy požadavků

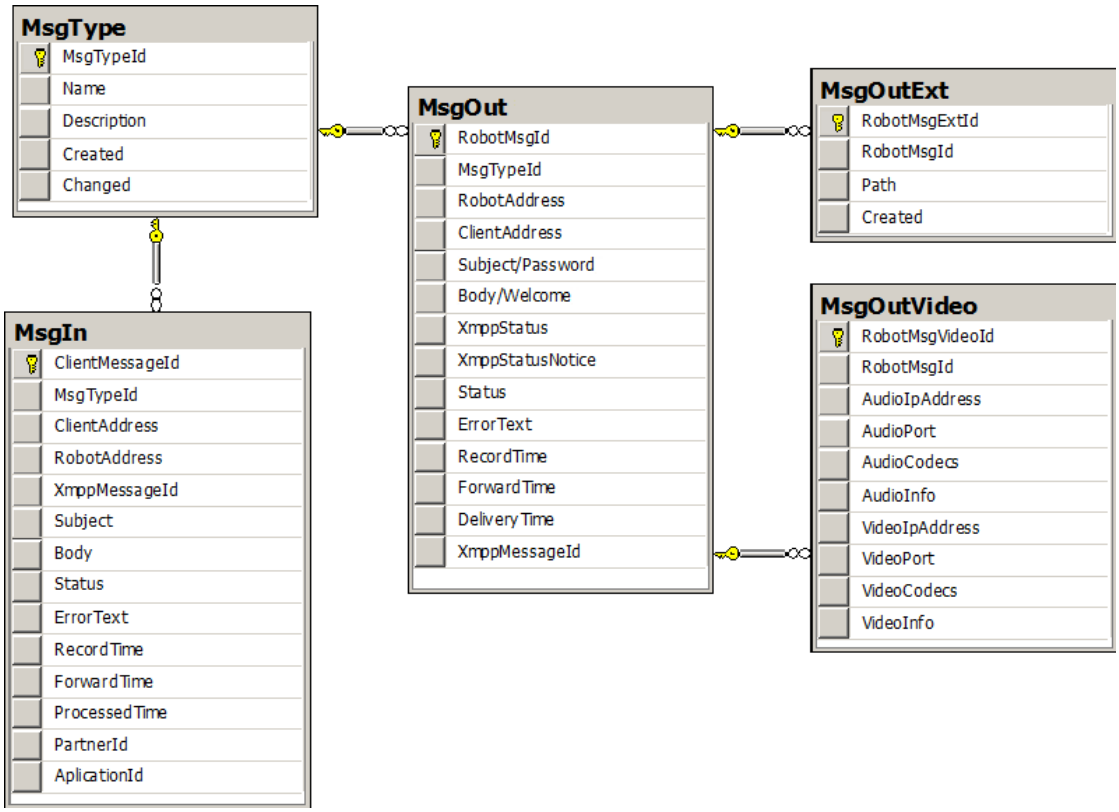
Existují pouze dvě výjimky. První je zpráva se soubory, kde může dojít k situaci, že k jedné zprávě bude náležet větší množství souborů, to by způsobilo nežádoucí redundanci na datech, proto tabulka obsahující názvy souborů byla od zbytku dotazů oddělena. Druhou výjimkou jsou videohovory, ty pro své obsloužení vyžadují velký počet povinných polí a také nejsou tak časté jako zbytek zadaných požadavků, což by způsobovalo velké množství prázdných polí. Navíc informace o videostreamech jsou v tuto chvíli navrženy tak, že k jednomu požadavku o videohovor náleží pouze jeden videostream a jeden audiostream, ale v budoucnu by mohlo dojít k rozšíření a k jednomu videohovoru by mohlo náležet více streamů, ze kterých by si koncový uživatel

Číslo	Stav
0	Čeká na obsloužení
1	Převzatý ke zpracování
2	Odeslaný klientovi
3	Doručený
996	Trvalá chyba – WCF
997	Trvalá chyba – Nutná reakce
998	Trvalá chyba
999	Opravitelná chyba

Tabulka 6: Vyvíjená aplikace – Fáze zpracování požadavku

vybral ten pro něj rychlejší a kvalitnější (více informací o videohovorech je uvedevo v kapitole 4.3.8).

Rozdělení na požadavky ze strany klienta a ze strany kamery má i ten důvod, že k oběma druhům požadavků jsou přiřazeny různé údaje pro správu. Obě varianty mají společné údaje: status zpracování, čas vložení, čas odeslání, id odeslaného XMPP dotazu (možnost dohledání) a případnou chybovou zprávu. Údaje od kamery, zpracovávané touto službou, obsahují navíc čas doručení a údaje od klienta obsahují čas zpracování a dva identifikátory aplikací společnosti Jablocom s. r. o., které je zpracovávají. Status zpracování je celé číslo identifikující daný postup ve zpracování (hodnoty využívané ve vyvíjené službě jsou uvedeny v tabulce číslo 6).



Obrázek 5: Schema databáze vyvíjené služby

4.2.1 WCF služba

V této kapitole budou popsány jednotlivé metody, které nabízí WCF služba pro přidávání a správu požadavků. V popisu budou používány termíny „jméno kamery“ to označuje XMPP adresu kamery, ovšem pouze její localpart, „výjimka“, tato výjimka je konkrétně představována typem určeným pro zasílání výjimek z WCF služeb tedy FaultException<ArgumentException> či „adresa klienta“, zde se rozumí bare JID z XMPP adresy klienta.

Před použitím služby WCF je nutno správně nastavit její parametry, kterými jsou: přístupové údaje do databáze vyvíjené služby, přístupové údaje do databáze XMPP serveru, doména XMPP serveru, formát automaticky vytvářeného hesla kamery a jméno skupiny, která na XMPP serveru obsahuje kamery (viz kap. 4.1.7 – nastavení XMPP serveru). WCF služba mimo hlavních metod obsahuje dvě pomocné, umožňující vytvoření administrátorského rozhraní. Metoda *GetXMPPserverDomain* navrací řetězec obsahující doménu zadaného XMPP serveru a metoda *CheckPassword*, která přejímá dva parametry jméno kamery a heslo a vrací hodnotu True, jestliže otisk hesla odpovídá otisku aktuálního hesla kamery. Pokud je jméno kamery prázdné, je vrácena výjimka. Hlavními metodami WCF služby jsou:

- *AddClient* – přidání adresy klienta do kontakt listu kamery. Přejímá tři parametry – jméno kamery (robotAcc), adresa klienta (clientAcc) a uvítací zpráva (wellcMsg). Uvítací zpráva není povinný parametr, pokud adresa klienta není bare JID nebo je jméno kamery prázdné je navrácena výjimka. Metoda vrací True, pokud byl požadavek úspěšně zapsán. Metoda také kontroluje aktuální stav účtu kamery, pokud je offline nebo ještě nebyl vytvořen, dojde k automatickému přidání požadavku o jeho spuštění.
- *RemoveClient* – odebrání adresy klienta z kontakt listu kamery. Přejímá dva parametry – jméno kamery (robotAcc) a adresu klienta (clientAcc). Jestliže je adresa v nesprávném formátu nebo je jméno kamery prázdné, dojde k vyhození výjimky. Metoda vrací True, pokud byl požadavek zapsán.
- *SendMessage* – odeslání zprávy na účet klienta. Přejímá čtyři parametry – jméno kamery (robotAcc), adresa klienta (clientAcc), subjekt zprávy (subject) a obsah zprávy (body). Jediný nepovinný parametr je subjekt, pokud bude jiný z parametrů prázdný řetězec či Null hodnota nebo adresa klienta nebude mít správný tvar, bude vyhozena výjimka. Metoda navrací hodnotu True, pokud byl požadavek úspěšně zapsán.
- *SendStatusChangeReq* – změna stavu kamery. Přejímá tři parametry – jméno kamery (robotAcc), XMPP status (statusXmpp) a poznámku (statusNotice). Poznámka není povinná. Pokud XMPP status nenabývá hodnoty Away, Chat, Dnd, Xa či NONE nebo je jméno kamery prázdné je vyhozena výjimka. Metoda navrací hodnotu True pokud byl požadavek úspěšně zapsán.
- *SendVideoReq* – přidání údajů o videohovoru. Přejímá jedenáct parametrů – jméno kamery (robotAcc), XMPP adresa klienta (clientAcc), informace zda

je videohovor iniciován klientem nebo kamerou (fromClient), IP adresa audiostreamu (aud_ip), port audiostreamu (aud_port), kodeky audiostreamu ve formátu „id,název,clockrate;id,...“ (aud_codecs), data audiostreamu ve formátu „audio_protokol,transportní_protokol,druh_streamu,jméno,heslo“ (aud_data), IP adresa videostreamu (vid_ip), port videostreamu (vid_port), kodeky videostreamu ve formátu „id,jméno,šířka,výška,framerate;id,...“ (vid_codecs) a data videostreamu ve formátu „video_protokol,transportní_protokol,druh_streamu,jméno,heslo“ (vid_data). Pokud je některý z parametrů v nesprávném formátu, dojde k vyhození výjimky, pokud byl požadavek úspěšně zapsán, vrátí metoda hodnotu True.

- *SendFileMsg* – odeslání zprávy se soubory. Přejímá pět parametrů – jméno kamery (robotAcc), adresa klienta (clientAcc), subjekt zprávy (subject), obsah zprávy (body) a pole souborů (fileNames). Subjekt zprávy je nepovinný. Pokud je některý z dalších parametrů prázdný nebo je prázdné pole se soubory, dojde k vyhození výjimky. Jestliže se požadavek o odeslání zprávy úspěšně zapsal, ale došlo k chybě při zapisování některého ze souborů, je požadavek v databázi označen statusem 996 a metoda vrací False, pokud dojde k úspěšnému zadání celého požadavku, vrací metoda True.
- *GetClientMsg* – metoda pro získání požadavků z tabulky zpráv od klienta. Přebírá osm parametrů – jméno kamery (robotAcc, nesmí být prázdné), max. počet (number, hodnota -1 nebo Null vrátí vše), typ požadavků (type, ID požadavku viz tab. 5, Null vrátí vše), interval statusů (statusFrom, statusTo, čísla viz tab. 6, hodnota Null odebere jednu/obě hranice), časový interval (dateFrom, dateTo, formát „měsíc.den.rok.hodina.minuta“, hodnota Null odebere jednu/obě hranice, adresa klienta (client, hodnota Null vrátí požadavky od všech klientů). Pokud je některý ze zadaných parametrů v nesprávném formátu je vyhozena výjimka. Jestliže během plnění požadavku dojde k chybě je vrácena hodnota Null, pokud vše proběhne v pořádku je vráceno pole objektů *ClientMessagesOfSelectedRobot* (obsahuje všechna pole klientské tabulky ve stejnojmenných properties).
- *GetRobotRequests* – metoda pro získání požadavků z tabulky zpráv od kamery. Přebírá osm parametrů – stejná definice jako u *GetClientMsg*. Pokud je některý z parametrů v nesprávném formátu, je vyhozena výjimka. Jestliže během plnění požadavku dojde k chybě, je vrácena hodnota Null, pokud vše proběhne

v pořádku, je vráceno pole objektů *RequestsToSelectedRobot* (obsahuje všechna pole tabulky robota ve stejnojmenných properties).

- *GetRobotState* – metoda vrací aktuální stav robota v závislosti na vykonaných požadavcích v databázi. Přebírá jediný parametr – jméno kamery (robotAcc), ten nesmí být prázdný, jinak dochází k vyhození výjimky. Metoda vrací celé číslo, kde -1 znamená neexistenci robota, 0 robot není spuštěn, 1 robot spuštěn a vykonává požadavky a -2 znamená chybu během vykonávání.
- *GetContacts* – metoda vracející obsah kontakt listu kamery. Přebírá jediný parametr – jméno robota (robot), ten nesmí být prázdný, jinak dochází k vyhození výjimky. Pokud během výkonávání dojde k chybě, je vrácena hodnota Null, v opačném případě pole objektů *ContactOfRobot* (obsahuje adresu klienta, jeho zkrácené jméno (nick) a stav oprávnění na stavovou informaci (subscription)).
- *GetRobotNames* – metoda pro získání jmen všech vytvořených kamer náležejících do vytvořené skupiny na XMPP serveru. Pokud dojde během vykonávání metody k chybě, je vrácena hodnota Null, v opačném případě je vráceno pole obsahující jména kamer jako řetězce.

4.3 Vlákno kamery

V následujících několika kapitolách budou popsány základní postupy při plnění požadavků zasílaných od kamery ke klientovi. Požadavky zasílané od klienta ke kameře jsou přijímány asynchronně a zpracovávány v jiném vlákně. Poté co takový požadavek dorazí na adresu kamery, dochází k jeho analýze a posouzení, tedy o který z požadavků se jedná, zda je požadavek adresován od uživatele s oprávněním (tj. uživatel, který se nachází v kontakt listu kamery) apod. Dále je automaticky zapsán do databáze této služby, odkud jej přebírá robot společnosti Jablocom s. r. o. a reaguje na něho dle stanovených postupů.

K naprogramování vlákna kamery byla použita knihovna agsXMPP, která je dostupná pod licencí GPL v2, což umožňuje její bezplatné použití v open source projektech. Knihovna je naprogramována v jazyce C# a je kompatibilní s technologiemi .NET či Mono. Výhodou knihovny je, že má implementovány základní mechanismy pro práci s XMPP technologií, jako je např. přihlašování k serveru (pomocí zabezpečeného i nezabezpečeného kanálu), resource binding a další. Také má implementovány základní XML stanza, které je tak možné snadněji sestavit. XML stanza, které implementovány nejsou lze odvodit od předvytvořených rozhraní. [12]

Pro vytvoření a spuštění vlákna kamery je nutné použít třídu *XMPPClientThread*. Konstruktor přejímá dva parametry, kterými jsou XMPP adresa kamery a heslo. Poté stačí pouze zavolat metodu Start, která nastartuje životní cyklus vlákna kamery (viz obrázek 6). Ve svém těle metoda start nastaví výchozí hodnoty všech proměnných a pomocí zmíněné knihovny provede připojení ke XMPP serveru. Ještě před připojením jsou také nastaveny základní parametry vlákna, jako je odeslání požadavku roster get (viz kap. 2.2.4), časový interval, po jehož uplynutí klient odešle prázdný XML stanza (informace pro server, že je XMPP klient vlákna stále aktivní, pouze aktuálně nekomunikuje se svým okolím), či callback funkce, které jsou aktivovány po příchodu vybraných XML stanza (IQ, message a presence).

Obrázek 6: Životní cyklus vlákna kamery

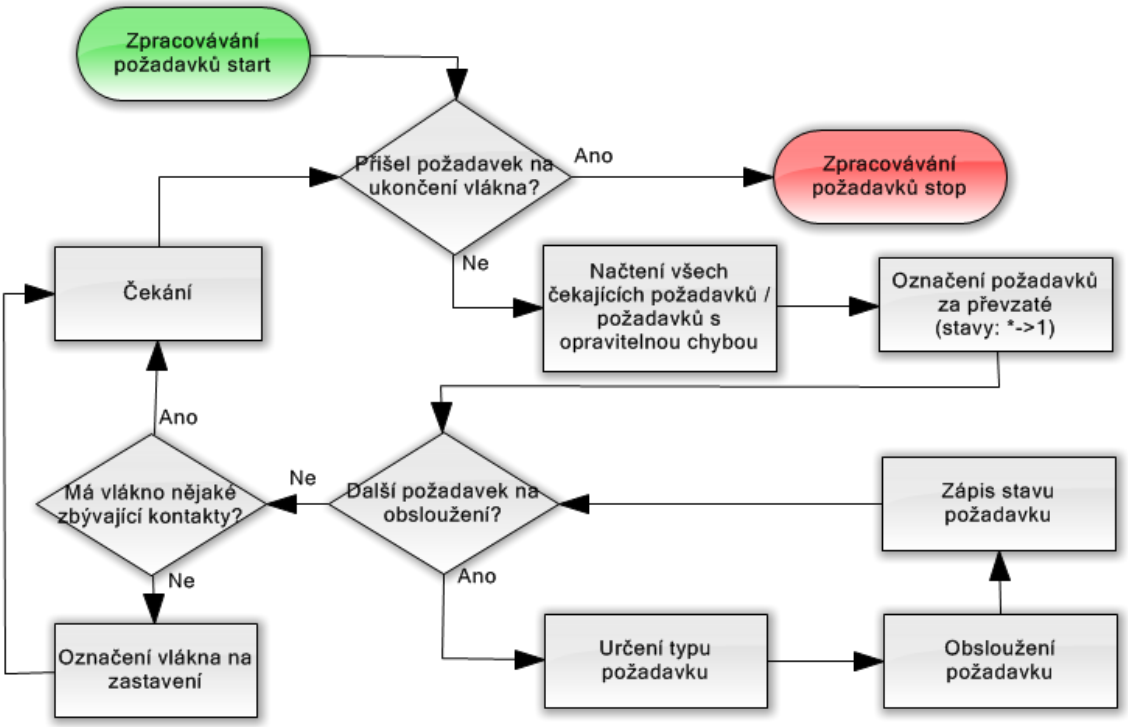
Dalším postupem je získání všech kontaktů, které má vlákno ve svém kontakt listu. Zde dochází také k automatickému filtrování kontaktů – pokud se v kontakt listu kamery vyskytne nějaký kontakt, který není kompletně spárován (subscription má jinou hodnotu než both), je automaticky z kontakt listu smazán. Hlavním důvodem je skutečnost, že pokud si kameru pokusí přidat uživatel, který k tomu není oprávněn, je jeho žádost ignorována, ale server Openfire kontakt ponechá v kontakt listu s subscription hodnoty none. Tím by v případě akceptování takového kontaktu mohlo dojít k nežádoucímu přidání kamery, proto jsou tyto nekompletní kontakty smazány, aby nemohlo dojít k narušení bezpečnosti kamery.

V další fázi je z databáze služby získán poslední stav, ve kterém se kamera nacházela (v případě, že není nalezen, dojde k použití výchozího stavu, který je nastaven v jednom z parametrů celé služby) a stav kamery je na něj změněn. Následně je nastartováno vlákno, které obsluhuje požadavky ze strany kamery (popis funkce viz kap. 4.3.1). Toto vlákno běží, dokud není doručen signál pro jeho ukončení (za-

volání metody Stop). Po zavolání této metody, dojde k dokončení aktuálního cyklu zpracovávání a následně je vlákno zastaveno. Po zastavení vlákna dojde k odhlášení XMPP klienta vlákna od serveru a kontrole, zda nezůstalo aktivní spojení s databází systému (pokud ano, je násilně ukončeno).

4.3.1 Zpracovávání požadavků

Po spuštění vlákna obsluhujícího požadavky od kamery dochází k periodickému provádění jeho základního cyklu, který je popsán na obrázku číslo 7. V první fázi dojde ke kontrole, zda nebyla zavolána metoda Stop a chod vlákna ukončen. Jestliže nikoli, dojde k načtení požadavků z databáze služby, toto je realizováno zavoláním vložené procedury, která je součástí databáze. Její součástí je i označení všech načtených požadavků jako převzatých ke zpracování (viz tab. 6). Načtení je možné upravovat v závislosti na parametrech zadaných službě, například zda se mají zasílat požadavky označené jako „odeslané“ (pouze ty u kterých je koncový stav „doručené“), „s opravitelnou chybou“ či „s trvalou chybou“. U všech typů je navíc možné stanovit časový interval, po jehož uplynutí dojde k znovuodeslání. Dalším parametrem je pak doba platnosti požadavku, tak aby se klientovi nedoručovaly informace, které již nejsou nějakou dobu platné.



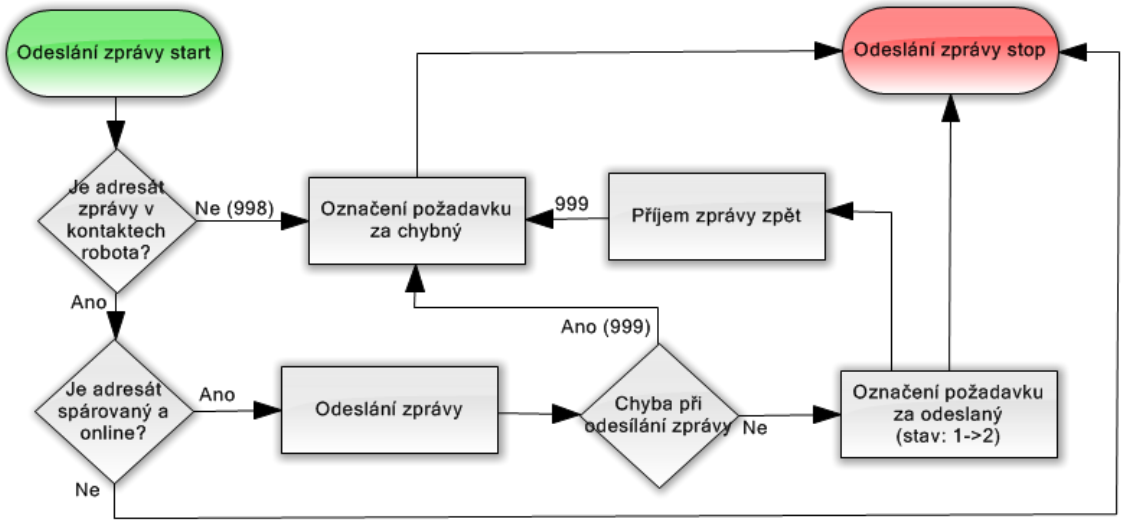
Obrázek 7: Vlákno kamery – Zpracování požadavků

Následně jsou požadavky obsluhovány v tom pořadí, v jakém byly vloženy do databáze služby. U všech dochází nejdříve k analýze (určení typu požadavku) a následně jsou příslušné parametry tohoto požadavku předány rutině pro jeho vykonání. Každá rutina po dokončení vrácí výsledek v podobě objektu *XMPPRequestResult*, který obsahuje informaci o tom, v jakém stavu se požadavek aktuálně nachází a další údaje, jako je chybové hlášení nebo XMPP ID odeslaného XML stanza. Aktuální stav je následně zapsán do databáze.

Na konci každého cyklu dochází k testu, zda se v kontakt listu kamery nachází alespoň jedna XMPP adresa. V případě že nikoliv zapíše vlákno automaticky požadavek na své ukončení. Zde je důležité, že i když je vlákno označeno k zastavení, nadále vykonává svou funkci – přijímá nové požadavky a o jeho ukončení rozhoduje až hlavní vlákno služby. Pokud by tak v následujícím cyklu došlo k načtení požadavku o přidání XMPP adresy klienta, vlákno ho automaticky obslouží a při dotazu od hlavního vlákna, zda má být vlákno skutečně ukončeno, dojde k jeho informování, že je vlákno nadále aktivní a nemělo by tudíž k jeho zastavení dojít. Poté přejde vlákno na předem určenou dobu (určena parametrem služby) do stavu spánku, po němž se cyklus opakuje.

4.3.2 Odeslání zprávy klientovi

Pokud na vlákno kamery dorazí požadavek na odeslání zprávy na IM klienta uživatele, dochází k jeho zpracování (viz obrázek č. 8).



Obrázek 8: Vlákno kamery – Odeslání zprávy

V první fázi dochází ke kontrole, zda se adresa uživatele nachází ve vnitřním seznamu kontaktů kamery (porovnání s databází služby by bylo neefektivní rychlostně a také existuje možnost, že si kameru z kontakt listu odebere sám uživatel). Jestliže adresa v kontakt listu není, dojde k označení požadavku trvalou chybou. Jestliže se adresa v kontaktech kamery nachází, dojde ke zjištění stavu uživatele, a pokud je uživatel ve stavu offline, je odeslání zprávy odloženo. V případě, že má uživatel online alespoň jeden resource, dojde k odeslání zprávy na Bare JID (odeslání na všechny resource uživatele).

Jestliže dojde k chybě během odesílání zprávy (např. chyba XMPP spojení) je požadavek označen opravitelnou chybou (zpráva může být odeslána znovu poté, co hlavní vlákno opraví chybu vlákna kamery). Stejně tak může dojít k situaci, kdy je zpráva vrácena zpět vláknu kamery (například výpadek serveru uživatele či server uživatele zprávu z nějakého důvodu odmítne). V tomto případě dojde k vyhledání ID požadavku v databázi a jeho označení opravitelnou chybou (nedostupnost serveru může být pouze dočasná).

4.3.3 Zabezpečení přenášených zpráv

Hlavní principy zabezpečení přenášených zpráv vycházejí z protokolu XMPP a nastavení použitého XMPP serveru. Tedy použití metody SASL k autorizaci obou účtů (klient i kamera) a použití zabezpečeného kanálu TLS pro přenos zpráv jak mezi službou kamery a domácím XMPP serverem, tak zejména mezi domácím XMPP serverem a XMPP serverem uživatele. Tyto mechanismy je díky otevřenosti této technologie možné rozšířit v podstatě o libovolný bezpečnostní mechanismus, například skrývání obsahu zprávy pomocí jeho vkládání do zdánlivě nesmyslného textu aj. Nevýhodou těchto řešení, je ovšem nutnost vytvořit XMPP klienta, který by danou techniku znal a dokázal s ní pracovat. Zároveň by tím byl uživatel „nucen“ ke stažení a používání specifického XMPP klienta, což není záměrem společnosti Jablocom s. r. o., která chce naopak uživatelům umožnit ovládání kamery z prostředí, na které jsou zvyklí.

Zabezpečení dat neznamená ovšem pouze zabezpečení samotného textu zprávy. Mezi bezpečnostní mechanismy, které je nutno diskutovat, patří i mechanismus zajišťující doručení zprávy klientovi, tedy opatření proti situaci, kdy odeslaná zpráva o poplachu v domě uživatele vůbec nebude doručena. Opět je možné využít vlastní řešení, ale XMPP protokol má pro tyto účely definována tři vlastní rozšíření: *Message*

Events (XEP-0022), *Advanced Message Processing (XEP-0079)* a *Message Delivery Receipts (XEP-0184)*.

Message Events je původním rozšířením protokolu XMPP s cílem zajistit stavovou informaci při zasílání zpráv mezi uživateli. Přidání elementu <x/> a dalších vnořených elementů, náležících do jmeného prostoru „jabber:x:event“, do těla odesílané zprávy umožňuje rozlišit čtyři události: Offline – signalizace toho, že byla zpráva uložena na XMPP serveru příjemce pro pozdější doručení, jelikož je příjemce aktuálně nedostupný, Delivered – zpráva byla doručena na XMPP klienta příjemce, Displayed – signalizace, že byla zpráva XMPP klientem zobrazena přímo příjemci a Composing – informace o tom, že příjemce zprávy píše odpověď na odeslanou zprávu. Ukázka odeslání požadavku o doručenku: [13]

```
<message to='klient4@example.cz' from='kamera2@jablocom.com' id='e4'>
  <body>Poplach!</body>
  <x xmlns='jabber:x:event'>
   <delivered/> <displayed/>
  </x>
</message>
```

Advanced Message Processing je rozšíření sloužící pro přidání pokročilé funkcionality zpracování zpráv do XMPP. Pomocí přidání elementu <amp/> a dalších elementů ze jmeného prostoru „http://jabber.org/protocol/amp“, umožňuje definování řady podmínek a akcí, které jsou aplikovány na odesílanou zprávu. Mezi akce patří odeslání potvrzení, odeslání chybového hlášení, zahození zprávy aj. Mezi definovanými podmínkami je pak vypršení platnosti zprávy, nedoručení zprávy, doručení zprávy konkrétnímu JID a řada dalších. Aplikace těchto pravidel probíhá na straně XMPP serverů, tudíž není vyžadována podpora na straně příjemce, ale pouze na jeho XMPP serveru. Ukázka požadavku o doručenku: [14]

```
<message from='kamera4@jablocom.com' id='u6' to='klient9@example.cz'>
  <body>Poplach!</body>
  <amp xmlns='http://jabber.org/protocol/amp'>
    <rule condition='match-resource' action='notify' value='any'/>
  </amp>
</message>
```

Poslední možností je rozšíření Message Delivery Receipts, které je zaměřeno přímo na potvrzování doručených zpráv. Rozšíření funguje díky vkládání elementů

<request/> a <received/> (patří do jmenného prostoru „urn:xmpp:receipts“) do odesílané respektive potvrzující zprávy. Na rozdíl od dříve zmíněného Advanced Message Processing, pracuje toto rozšíření na klientské úrovni, což je pro odesílání doručenek vhodnější. Ukázka odeslání žádosti o doručenku ve zprávě: [15]

```
<message id='v' from='kamera3@jablocom.com' to='kli7@example.cz/res'>
  <body>Poplach!</body>
  <request xmlns='urn:xmpp:receipts'/>
</message>
```

Obsluha, tedy odesílání požadavku i příjímání doručenky všech zmíněných variant, byla do služby zabudována (lze je zapnout během překladu programu definováním symbolů ME, AMP či MDR), ovšem žádná z nich není ve výchozím stavu aktivní, jelikož všechna tato rozšíření trpí určitými nedostatky. Například rozšíření Message Events bylo označeno jako zastaralé, tudíž jeho implementování do XMPP klientů není doporučeno. [13] Největším problémem těchto rozšíření je ovšem jejich omezená podpora jak na straně XMPP serverů, tak na straně XMPP klientů. V tabulce číslo 4 si lze prohlédnout, jak jsou na tom s podporou volně dostupné XMPP servery (plně podporováno je pouze Message Delivery Receipts, a to jen díky tomu, že server zprávy tohoto typu pouze přeposílá, nemusí tedy jejich obsahu rozumět). Větším problémem je ovšem podpora na straně XMPP serverů třetích stran, které poskytují největší základnu potenciálních uživatelů. Například XMPP server společnosti Google Inc. nepodporuje žádnou ze zmíněných variant, viz seznam podporovaných jmenných prostorů (rozšíření), získaný odesláním příslušného požadavku:

```
<iq xmlns="jabber:client" to="ponikelsky.jakub@gmail.com/res"
    id="disco1" type="result" from="gmail.com">
 <query xmlns="http://jabber.org/protocol/disco#info">
  <identity category="server" type="im" name="Google Talk" />
  <feature var="http://jabber.org/protocol/disco#info" />
  <feature var="google:jingleinfo" />
  <feature var="google:roster" />
  <feature var="google:nosave" />
  <feature var="google:setting" />
  <feature var="google:shared-status" />
  <feature var="http://jabber.org/protocol/archive#otr" />
  <feature var="google:queue" /><feature var="google:mail:notify" />
```

```
  <feature var="http://jabber.org/protocol/archive#save" />
 </query>
</iq>
```

Stejně tak omezená je i podpora na straně XMPP klientů. Navíc značná množina XMPP klientů, kteří tato rozšíření podporují, je má ve výchozím stavu vypnutá nebo k jejich funkci potřebuje doinstalovat zvláštní plugin. To by v praxi znamenalo, že by společnost Jablocom s. r. o. musela svým klientům doporučit jiný XMPP klient, než který aktuálně používají, což by vedlo k menšímu zájmu o výslednou službu. Následující dotaz zobrazuje podporu jmenných prostorů XMPP klienta Google Chat, získanou odesláním příslušného požadavku:

```
<iq xmlns="jabber:client" to="ponikelsky.jakub@gmail.com/res"
    id="disco2" type="result" from="ponikelsky_test@gmail.com/res">
 <query xmlns="http://jabber.org/protocol/disco#info">
  <identity category="client" type="pc" />
  <feature var="http://jabber.org/protocol/disco#info" />
 </query>
</iq>
```

Výsledkem této nízké podpory by byl fakt, že by stav většiny požadavků o zaslání zprávy zůstal na hodnotě „odeslaný“. Z tohoto důvodu byl tento stav u požadavku o odeslání zprávy označen jako koncový a potvrzující doručenka není společností Jablocom vyžadována. A to je důvod proč jsou všechna zmíněná rozšíření ve výchozím stavu vypnuta.

4.3.4 Změna stavu kamery

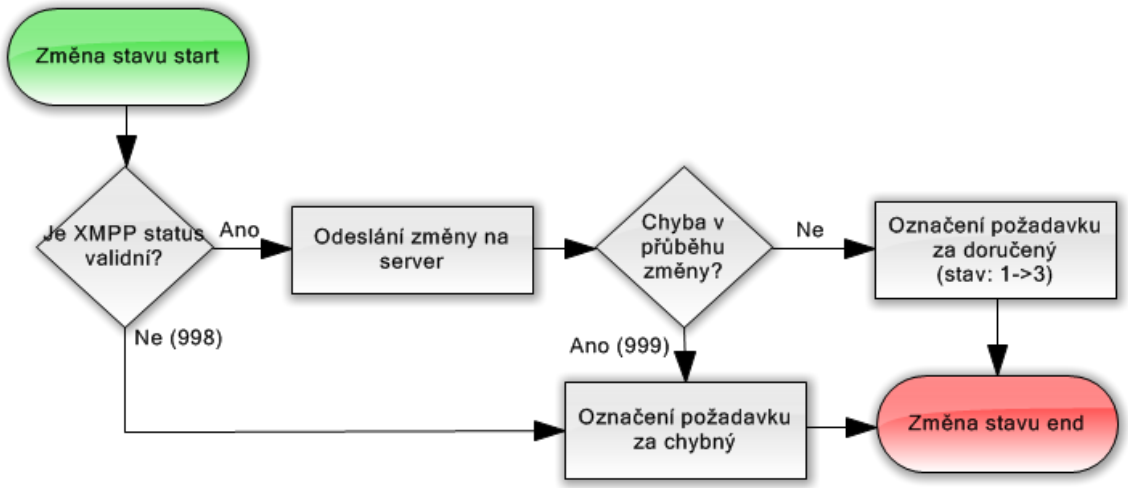
Změna stavu kamery je velmi jednoduchý proces (obr. č. 9), jehož jedinou součástí je sestavení elementu <presence/> s vybranými vlastnostmi. Před jeho sestavením je provedena pouze kontrola příchozího XMPP stavu. Ten musí nabýt jedné z hodnot „NONE“, „chat“, „away“, „dnd“ a „xa“. Jestliže požadavek nabyl jiné hodnoty, je označen trvalou chybou. Dále je do elementu přidán textový popis stavu, není povinnou součástí, takže může zůstat prázdný. Priorita tohoto stavu je stanovena na pět (standardní hodnota je nula), takže stav kamery by měl být zvýhodněn v případě, že by na účet byly připojeny naráz dva resource.

Před odesláním je do elementu přidán ještě element <c/>, který slouží k informování uživatele používajícího XMPP klienta Google Talk o tom, že služba kamery

umí přijímat a odesílat videohovory (více o videohovorech viz kapitola 4.3.8). Tento element musí být přidán ke každému stavu, jelikož pokud není jeho součástí Google Talk / Google Chat, zablokuje možnost volání ze strany uživatele na kameru. Podoba elementu <c/>: [16]

```
<caps:c xmlns:caps="http://jabber.org/protocol/caps"
   node="http://mail.google.com/xmpp/client/caps" ver="1.1"
   ext="voice-v1 video-v1 camera-v1" />
```

Následně je element <presence/> odeslán XMPP serveru. Pokud během odesílání dojde k chybě, je požadavek označen opravitelnou chybou a pokus o jeho zaslání by se měl opakovat. V opačném případě je požadavek označen za doručený, jelikož po doručení elementu na XMPP server dojde k jeho automatickému přeposlání všem kontaktům kamery.

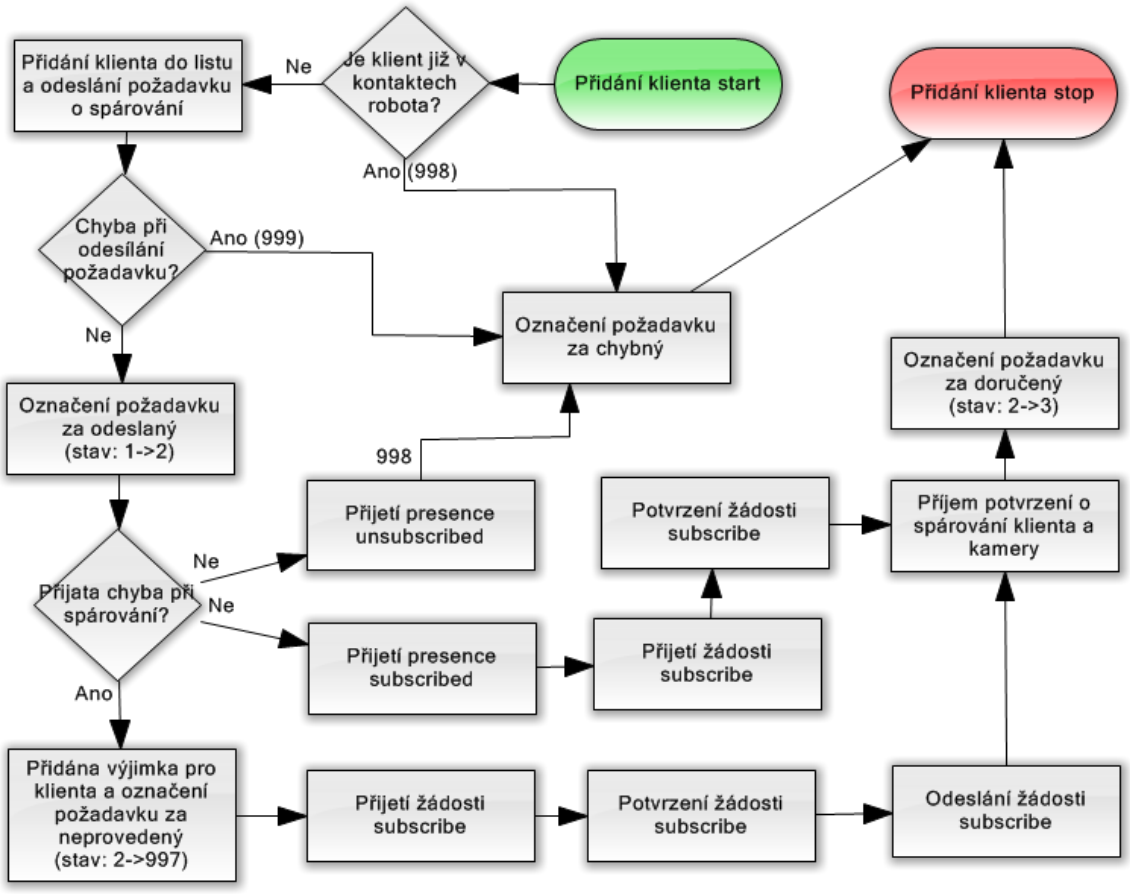


Obrázek 9: Vlákno kamery – Změna stavu

4.3.5 Přidání XMPP adresy do kontakt listu

Zjednodušený postup přidání XMPP adresy uživatele do kontakt listu kamery je zobrazen na obrázku číslo 10. První fází je kontrola, zda se daná XMPP adresa již v kontaktech kamery nenachází, a pokud ano, je požadavek označen trvalou chybou. Pokud se v kontaktech kamery nachází, ale nebylo ještě kompletně provedeno spárování, dochází k znovuodeslání požadavku subscribe (viz kap. 2.2.6) a znovuoznačení požadavku za odeslaný. Jestliže se adresa v kontaktech kamery nenachází, je sestaven požadavek roster set (viz kap. 2.2.4), jehož odesláním na XMPP server je adresa přidána do kontaktů kamery na XMPP serveru. Současně je také adresa

přidána do vnitřního kontakt listu vlákna kamery a také je odeslán požadavek subscribe o povolení stavové informace mezi klientem a kamerou. Pokud v kterékoli fázi této inicializace dojde k chybě, je požadavek označen opravitelnou chybou, v opačném případě je označen jako odeslaný.



Obrázek 10: Vlákno kamery – Přidání adresy klienta do kontakt listu

V další fázi se čeká na odpověď uživatele. Jestliže dojde k doručení elementu <presence/> typu unsubscribed, znamená to, že uživatel spárování s kamerou odmítl. V tomto případě je požadavek označen trvalou chybou, aby nedošlo k znovuzasílání žádosti a případnému spamu. Zároveň je také XMPP adresa klienta odebrána z kontaktů kamery jak ve vnitřním kontakt listu, tak v kontakt listu na XMPP serveru.

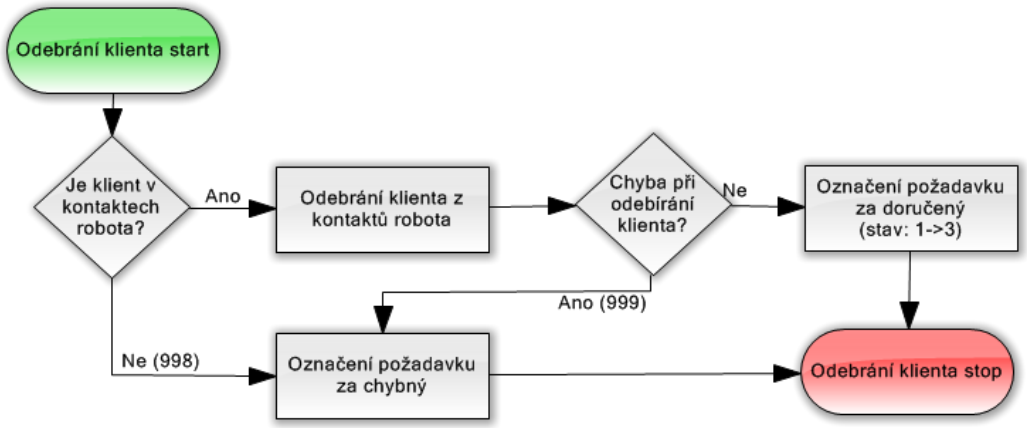
Jestliže dojde k doručení elementu <presence/> typu error, znamená to, že požadavek byl zamítnut některým z XMPP serverů, které ho doručují. K takové situaci dochází jestliže má XMPP server uživatele například zakázáno doručování subscribe dotazů z externích domén. V tomto případě je zpravidla nutné, aby spárování kamery a uživatele inicioval sám uživatel. Ve vnitřním kontakt listu kamery dojde k udělení výjimky (v běžné situaci není možné, aby spárování inicioval klient, kvůli narušení

bezpečnosti) a označení požadavku opravitelnou chybou, na kterou je nutná reakce ze strany společnosti Jablocom s. r. o. (tak, aby mohl být uživatel informován, že je nutná jeho spolupráce). Dále vlákno kamery čeká na doručení žádosti subscribe od uživatele. Po jejím doručení je žádost akceptována a odeslána vlastní žádost o povolení stavové informace (ta by již měla dorazit k posouzení uživateli, jelikož má kameru přidanou ve svém kontakt listu).

Poslední možností je, že dojde ke standardnímu procesu spárování a klient potvrdí požadavek odesláním <presence/> typu subscribed. V tomto případě se vyčká na přijetí žádosti subscribe od klienta, která je potvrzena. Jestliže dojde ke správnému spárování účtu klienta i kamery, dorazí na účet kamery element <iq/> typu roster set, s XMPP adresou uživatele a atributem subscription hodnoty both. V tomto okamžiku dojde k označení původního požadavku za doručený.

4.3.6 Odebrání XMPP adresy z kontakt listu

Na rozdíl od požadavku o přidání XMPP adresy uživatele do kontaktů kamery, je odebrání této adresy z kontakt listu relativně snadná operace (viz obr. č. 11). K obsloužení stačí pouze sestavit příslušný element <iq/> typu roster set a do něj vložit element <item/> s parametrem subscribtion hodnoty remove a bare JID uživatele (viz kap. 2.2.6). Po odeslání požadavku XMPP server automaticky odešle presence unsubscribe, unsubscribed nebo oba a tím zajistí i odebrání povolení na stavovou informaci.



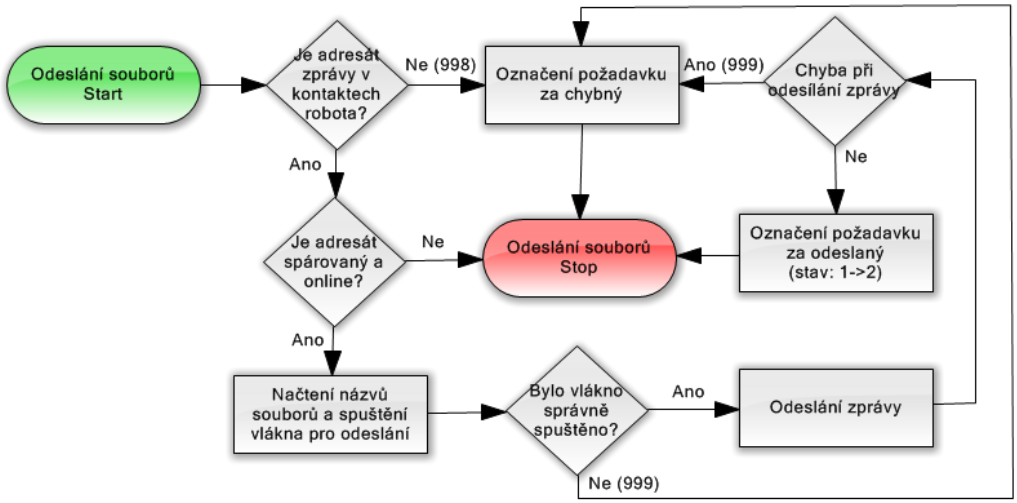
Obrázek 11: Vlákno kamery – Odebrání adresy klienta z kontakt listu

Před odesláním samotného požadavku dochází pouze ke kontrole, zda se uživatel nachází v kontaktech kamery, a pokud ne, je požadavek označen trvalou chybou, jelikož byl uživatel odebrán již dříve (či přidán vůbec nebyl). Následně je sestaven příslušný element a ten je odeslán na XMPP server – pokud by v průběhu odesílání došlo k chybě,

je požadavek označen opravitelnou chybou. V opačném případě je požadavek označen za doručený, jelikož o další postup se stará sám XMPP server.

4.3.7 Odeslání zprávy se soubory

Požadavek na odeslání zprávy se soubory je rozdělen na dvě části. První je velmi podobná principu odeslání samotné zprávy (viz obr. č. 12), ve druhé části pak obsluhu odesílání souboru přebírá samostatné vlákno, které přenos souboru/souborů uživateli dokončí (jeho funkce viz obr. č. 13).



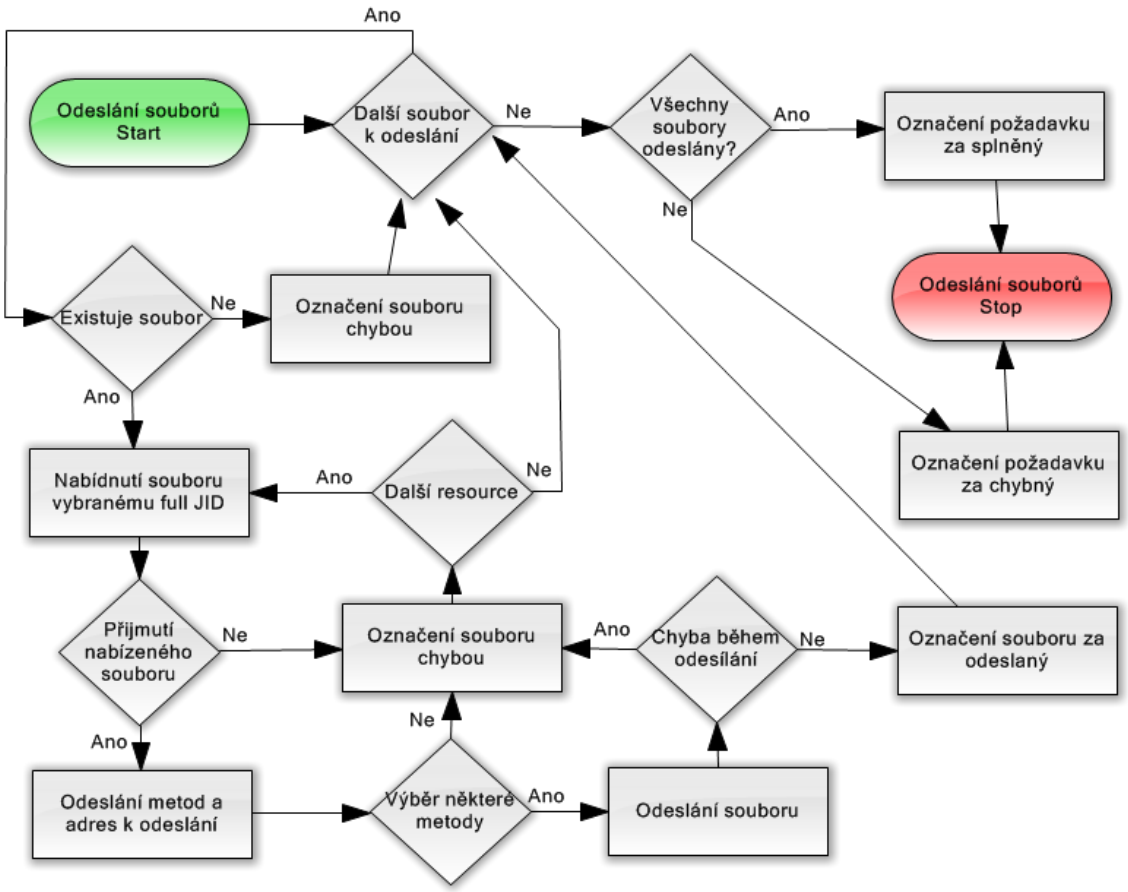
Obrázek 12: Vlákno kamery – Odeslání zprávy se soubory

V první části dochází, stejně jako u odeslání zprávy, ke kontrole dostupnosti uživatele. Tedy zda se XMPP adresa uživatele nachází ve vnitřním kontakt listu kamery, a pokud ne, je požadavek označen trvalou chybou. V případě, že ano, je ověřena jeho dostupnost. Pokud je online alespoň jeden z uživatelových resource, dochází k dalšímu zpracování požadavku, v opačném případě je požadavek odložen do doby, než bude uživatel dostupný.

Následující činností je načtení všech souborů, které k požadavku náleží, a vytvoření objektu *XMPPClientFileSender*, kterému jsou předány všechny potřebné parametry (informace o požadavku – adresa uživatele, jméno robota, seznam dostupných resource uživatele, seznam souborů a jiné). Následuje otestování seznamu příchozích souborů, který musí obsahovat alespoň jeden soubor, jinak je požadavek označen trvalou chybou. Jestliže obsahuje jeden a více souborů, dochází k zavolání metody Start zmíněného objektu, která spustí vykonávání vlákna odesílajícího soubory. Jestliže se spuštění vlákna nezdaří, je požadavek označen opravitelnou chybou. V opačném

případě je aktivována rutina sloužící k odeslání textového doprovodu, která je popsána v kapitole 4.3.2 (operace, které se duplikují s vykonáváním této rutiny, se znovu neprovádějí).

Vlákno, spravující odeslání všech souborů přiložených ke zprávě, si nejdříve během inicializační části převede seznam souborů na seznam objektů *FileToSendClass*, které obsahují parametry důležité k obsloužení požadavku (název souboru, id souboru, stav souboru). Po spuštění pak aktivuje svůj životní cyklus popsaný na obrázku č. 13. To znamená, že vybere první soubor, který se v seznamu nachází, a ověří jeho existenci. Pokud soubor není nalezen, automaticky dochází k označení jeho stavu jako chybového a přechodu na další soubor v pořadí.



Obrázek 13: Vlákno kamery – Vlákno odesílající soubory

Jestliže soubor existuje, dojde k sestavení full JID uživatele (předané bare JID a první ze seznamu resourcepart). Následně je také sestaven element <iq/>, obsahující element <si/> náležící jmennému prostoru „http://jabber.org/protocol/si/profile/file-transfer“, který nese údaje o souboru jako je jeho název, velikost, metoda použitá k jeho přenosu (zde zvolená metoda je SOCKS5 Bytestream ze jmenného prostoru

„http://jabber.org/protocol/bytestreams“) a další. Tento element je následně odeslán na sestavené full JID uživatele a čeká se na jeho odpověď.

Pokud je vrácena odpověď typu error (což znamená, že XMPP klient uživatele nepodporuje přenos souborů pomocí rozšíření SI File Transfer) či přichází informace o tom, že XMPP klient nepodporuje přenosovou metodu SOCKS5 Bytestream, dochází k sestavení dalšího full JID uživatele (pokud je dostupných více jeho resource) a element <iq/> s informacemi o souboru je odeslán na tuto adresu. Takto se postupuje, dokud některý z resource uživatele nepřijme nabídku pro odeslání souboru nebo dokud nejsou vyčerpány všechny adresy uživatele. Pokud jsou vyčerpány všechny adresy uživatele a soubor se nepodařilo uživateli odeslat, je stav souboru označen jako chybový a na řadu přichází další soubor ze seznamu. Jelikož ovšem žádný z resource nepodporuje přenos souborů vybranou metodou, dochází k následnému označení všech stavů souborů jako chybových (automaticky, bez dalších pokusů o odeslání souboru).

Jestliže vybrané resource uživatele odpoví <iq/> typu result s oznámením o podpoře obou technologií, přechází vlákno do další fáze odesílání souboru. Zde dochází k sestavení elementu <iq/> a podelementu <query/>, náležícímu jmennému prostoru „http://jabber.org/protocol/bytestreams“. Tento element nese informace o nabízených adresách, na kterých bude soubor dostupný. Zde je možné využít dvou variant přenosu souboru (přímý přenos ze serveru společnosti Jablocom s. r. o. na XMPP klienta uživatele či využití proxy jako prostředníka). Obě tyto možnosti je možné nastavit pomocí parametrů předávaných hlavní službě (povolení přímé metody, povolení proxy, adresa proxy serveru, port proxy serveru, IP proxy serveru, IP adresa počítače pro přímou metodu dostupná z vnější sítě, port počítače, na kterém bude soubor nabízen). Jestliže je povolena přímá metoda přenosu, je také vytvořen objekt typu *JEP65Socket* (tento objekt je součástí knihovny agsXMPP, respektive jeho použití je možné najít v ukázkovém kódu aplikace MiniClient [12]), který naslouchá na zvoleném portu. V případě, že dojde ze strany uživatele k navázání spojení, je mu odpovězeno tak, aby toto spojení mohlo být použito pro odeslání vybraného souboru. Jasnou nevýhodou přímé metody odesílání souborů je skutečnost, že lze naráz odesílat pouze jeden soubor, což při existenci více účtů kamer bude znamenat omezené možnosti zasílání souborů (nutnost žádosti zřetězit). Lepší variantou je tak přenos souboru přes proxy server, který je dostupný jako součást serveru Openfire. Následně je sestavený element <iq/> odeslán uživateli a čeká se na jeho odpověď.

Jestliže přijde jiná odpověď než typu result (jelikož například nebylo navázáno spojení pomocí přímé metody), dochází k pokusu o odeslání souboru jinému full JID uživatele. Pokud byla pro přenos souboru vybrána přímá metoda, dochází k samotnému odeslání souboru pomocí vytvořeného spojení. Jestliže byla vybrána metoda pomocí proxy serveru, dochází nejdříve k vytvoření spojení s proxy serverem, opět pomocí objektu *JEP65Socket*, a následně odeslání aktivačních elementů, které obsahují i full JID uživatele a další nutné informace pro spárování se spojením od uživatele. Pomocí tohoto spojení je pak samotný soubor odeslán. Pokud se kdykoli během odesílání souboru vyskytne chyba, dochází k pokusu o jeho zaslání jinému full JID uživatele. Pokud se naopak odeslání souboru podaří, je soubor označen jako odeslaný a dochází k pokusu o odeslání dalšího souboru stejnému full JID.

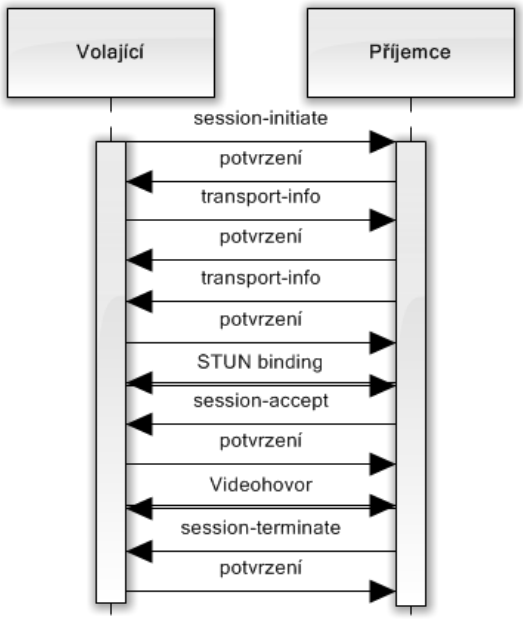
Na závěr, když dojde k pokusu o odeslání všech souborů v seznamu, přichází poslední část funkce vlákna. V této části je otestován stav všech souborů v seznamu. Jestliže je některý z nich označen jako chybový, je celý požadavek o odeslání zprávy se soubory označen trvalou chybou a součástí popisu této chyby je seznam všech souborů, které se nepodařilo odeslat. Pokud jsou všechny soubory v seznamu označeny jako odeslané, je celý požadavek označen jako doručený.

4.3.8 Přenos videa protokolem XMPP

Přenos videohovorů protokolem XMPP je realizován pomocí rozšíření s označením Jingle, zejména pak Jingle RTP Sessions (XEP-0167), které definuje přesné postupy při navazování videohovoru. Společnost Google Inc., na jejíž produkty je tato služba zaměřena nejvíce (jelikož má velkou členskou základnu), má ovšem přenos videohovoru realizován pomocí poněkud upravené varianty tohoto rozšíření. Přesný postup, jak navázat videohovor s produkty společnosti Google Inc. je popsán v dokumentu dostupném na https://developers.google.com/talk/call_signaling. Hlavní rozdíl spočívá v tom, že rozšíření XEP-0167 nejdříve naváže audio spojení a až posléze je do tohoto spojení přidán další obsah, kterým je video. Liší se také obsah některých zasílaných elementů, což ve výsledku zjednodušuje celý proces navazování videohovorů. Do této služby je tedy implementován postup navazování společnosti Google Inc. tak, aby byly videohovory dostupné pro klienty Google Talk / Google Chat. [17] [16]

Zákl. princip navázání videohovoru s XMPP klienty společnosti Google Inc. je popsán na obrázku číslo 14. Navazování je založeno na čtyřech základních elementech, které je nutné do služby implementovat (samotná knihovna agsXMPP nepod-

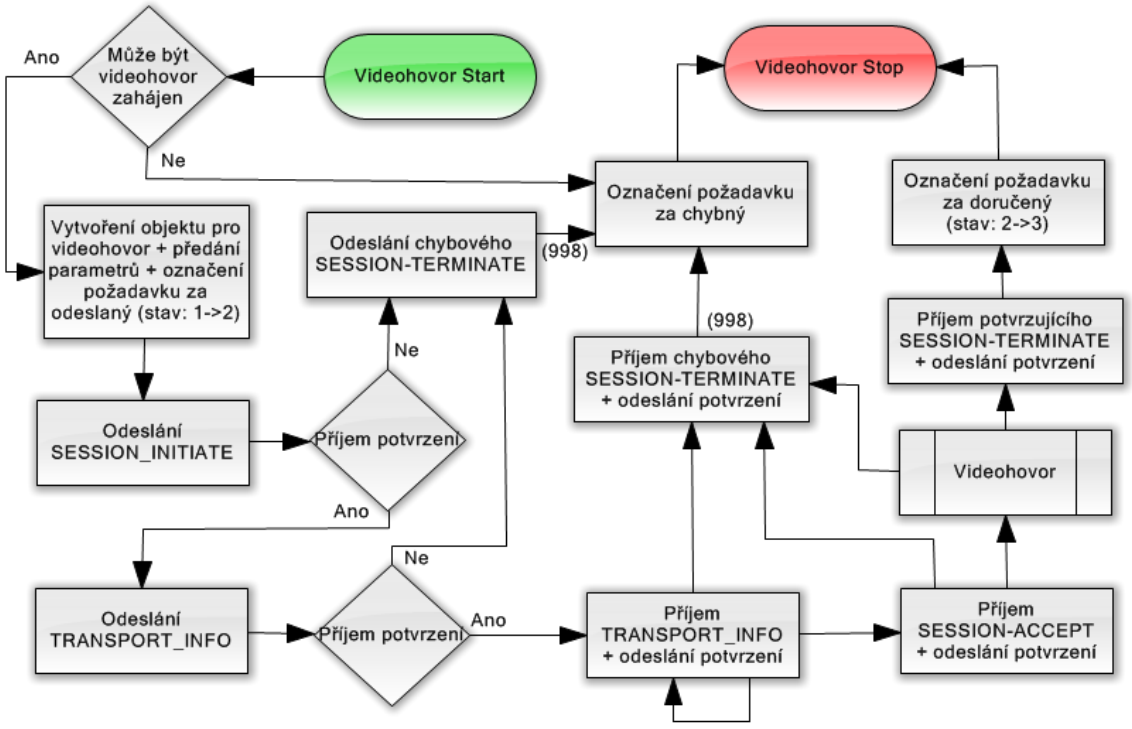
poruje rozšíření XEP-0167, ani řešení Google Inc.). Jsou jimi: session-initiate, který obsahuje základní informace o podporovaných kodecích na straně volajícího, pomocí kterých by měl být audio i videohovor kódován. Transport-info, který nese informaci o samotném videostreamu (tzn. IP adresa, port, protokol TCP/UDP, protokol RTP/RTCP, typ streamu (local, stun, relay, ...), bezpečnostní jméno a heslo a další údaje), opět odděleně pro audio i video část. Session-accept, kterým je hovor přijat a který obsahuje informace o kodecích, které podporuje volající i příjemce a pomocí kterých bude výsledný videohovor uskutečněn. Posledním typem elementu je pak session-terminate, kterým je videohovor ukončen a který obsahuje informaci o tom, proč byl videohovor ukončen (například informaci o chybě hovoru, chybě spojení nebo položku success, pokud videohovor proběhl a jeden z volajících pouze zavěsil). [16]



Obrázek 14: Princip videohovoru

Po konzultaci se společností Jablocom s. r. o. bylo rozhodnuto, že by bylo vhodné, aby videohovor mohl začít jak ze strany kamery, tak zejména ze strany uživatele. První variantou je tedy zahájení videohovoru ze strany kamery (postup je vidět na obrázku číslo 15). První částí postupu je zjištění, zda vůbec může být videohovor zahájen, zde dochází k testu, zda je videohovor určen uživateli nacházejícímu se v kontakt listu kamery (pokud ne, požadavek je označen trvalou chybou), zda již aktuálně neběží jiný videohovor s uživatelem (pokud ano, požadavek je označen opravitelnou chybou) a dále, zda je k dispozici nějaký resource uživatele, který tento videohovor podporuje (tzn., dříve byl doručen jeho stavový údaj obsahující navíc element <c/> s informací o podpoře videa a audiohovoru (viz kapitola 4.3.4)). Pokud takový resource k dispozici není, je videohovor odložen. V opačném případě dochází k testu dat, která byla spolu s požadavkem zapsána do databáze (nedochází k ověření po stránce funkcionální (tedy např. zda zadaná IP adresa opravdu existuje a stre-

amuje videohovor), ale po stránce syntaktické (tzn., zda je IP adresa skutečně IP adresa či zda mají zadané kodeky požadovaný tvar)). Pokud nejsou data ve správném formátu, je požadavek označen trvalou chybou. V opačném případě dochází k vytvoření objektu *VideoCallSessionFromRobot* a předání všech požadovaných parametrů (bare JID uživatele, podporující resource, video data atd.). Tento objekt je pak přidán do seznamu aktuálně probíhajících videohovorů (seznam objektů *AbstractVideoCallSession* – abstraktní třída pro oba typy videohovorů).



Obrázek 15: Vlákno kamery – Videohovor od kamery

Po vytvoření tohoto objektu a předání všech dat objekt automaticky vyšle element session-initiate s údaji o podporovaných kodecích a čeká se na odpověď. Jestliže dojde odpověď jiného typu než result (potvrzení) a nebyl předtím videohovor ukončen zasláním session-terminate ze strany klienta, dojde k ukončení hovoru ze strany kamery s odůvodněním cancel a označení požadavku o videohovor trvalou chybou. Pokud dorazí odpověď typu result, je fáze session initiate označena za úspěšnou, dojde k automatickému odeslání elementu typu transport-info s údaji o streamu (IP adresa, port ...) a čeká se na odpověď. Pokud dorazí odpověď typu result, je fáze transport-info označena za úspěšnou. V opačném případě dojde k ukončení videohovoru.

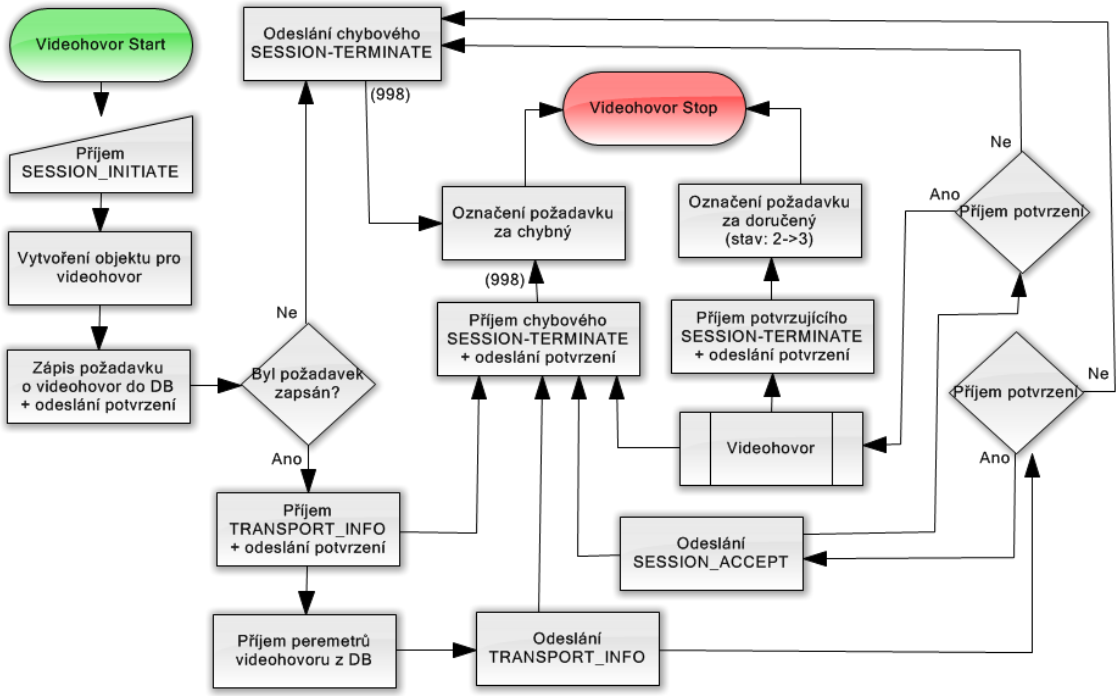
Asynchronně s těmito odesílanými elementy přicházejí elementy ze strany uživatele přijímajícího videohovor. Po doručení požadavku typu transport-info (tento

požadavek může být doručen i několikrát), je sestaveno příslušné potvrzení a to je odesláno uživateli. Zároveň také dojde k označení této fáze za splněnou – po doručení a potvrzení tohoto elementu (z obou stran), by mělo dojít k ověření funkčnosti daného streamu pomocí STUN binding requestu z obou stran (odeslání STUN requestu neprobíhá z této služby, ale přímo z počítače, ze kterého je vysílán videostream, jinak nelze obě strany hovoru spárovat) a pokud dojde ke spárování, je možné videohovor zahájit. Toto klient provede zasláním elementu session-accept, na který mu kamera sestaví potvrzení a označí videohovor za zahájený. Posledním typem elementu, který může být doručen, je session-terminate, po kterém dochází také ke zformulování příslušného potvrzení, je ovšem nutné vyhodnotit, zda byl požadavek o videohovor úspěšně proveden či nikoli. Toto je provedeno jednak pomocí obsahu elementu (měl by obsahovat položku success) a také pomocí označení splněných fází (měly by být splněny všechny). Na základě těchto údajů je požadavek označen buď za doručený, nebo trvalou chybou.

Druhou možností je zahájení videohovoru ze strany klienta (viz obr. č. 16). Jestliže na vlákno kamery dorazí element <iq/> s podelementem <jingle/> typu session-initiate, od známého uživatele, dochází k automatickému vytvoření objektu *VideoCallSessionFromClient* a jeho přidání mezi aktivní videohovory. Tento objekt sestaví příslušné potvrzení, označí fázi session-initate za splněnou a hlavně zapíše do databáze služby požadavek typu *RequestVideoClient*, který znamená požadavek o audio i video stream. Jestliže se požadavek o stream nepodaří zapsat (např. výpadek spojení s databází), je videohovor ukončen ze strany kamery. V opačném případě dochází k automatickému potvrzování elementů transport-info ze strany uživatele a čekání na příchozí informace o streamu.

V okamžiku, kdy je do databáze služby zapsán požadavek typu *VideoClient* s daty o streamu, dochází k ověření existence videohovoru (zda nebyl dříve ukončen) a testu platnosti přijatých dat. Jestliže některá z podmínek není splněna, je požadavek označen trvalou chybou. V opačném případě jsou data předána objektu. Ten automaticky odešle element transport-info a čeká na jeho odpověď – v tomto okamžiku by mělo dojít ke spárování STUN requesty – pokud se spárování povede, uživatel odešle potvrzení elementu transport-info. Po jeho přijetí vlákno kamery automaticky zašle element session-accept, po jehož potvrzení by měl být videohovor odstartován. Pokud na kterýkoli z těchto dvou elementů přijde od uživatele jiné než kladné potvrzení, dojde k ukončení videohovoru ze strany kamery. V okamžiku, kdy na vlákno kamery

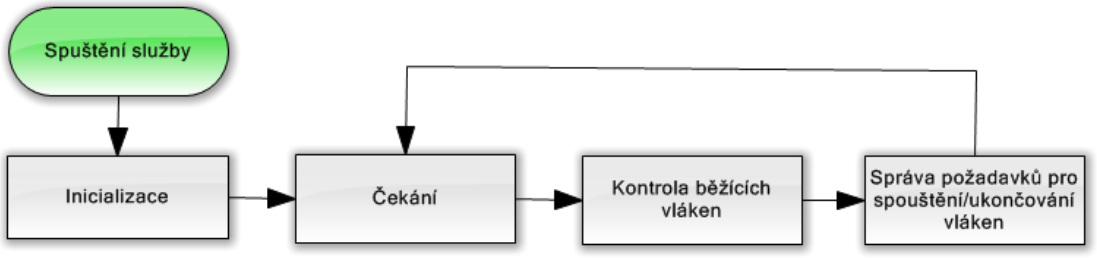
dorazí element session-terminate, dojde k jeho vyhodnocení na základě stejných kritérií jako u videohovoru ze strany kamery (obsah a označení splněných fází) a příslušnému označení požadavku *VideoClient* za doručený či s trvalou chybou.



Obrázek 16: Vlákno kamery – Videohovor od klienta

4.4 Hlavní služba

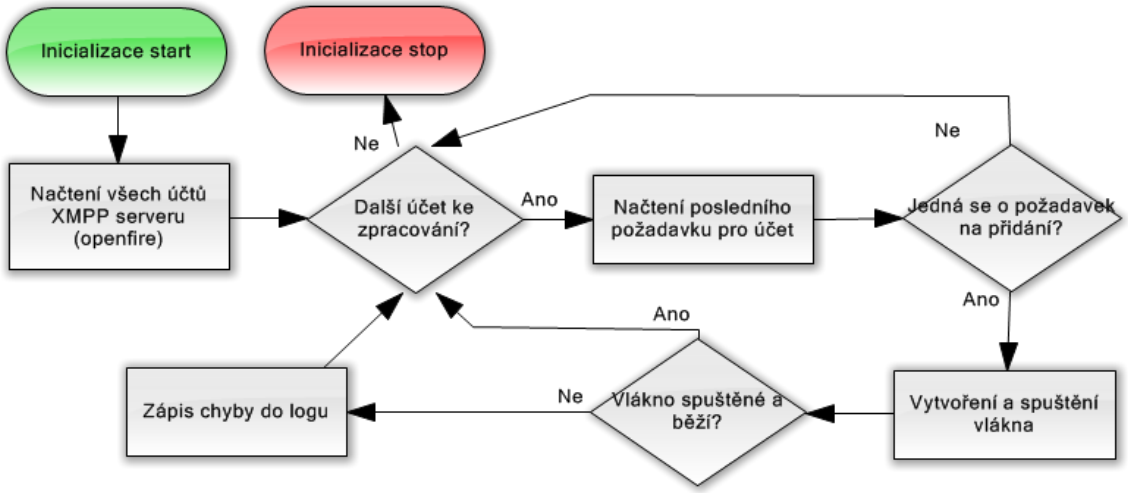
Hlavní služba (obecné schéma viz obrázek číslo 17) je oddělené vlákno, které se stará o běh všech vláken jednotlivých kamer. Skládá se ze tří základních operací: inicializace po startu služby, při které musí být spuštěna všechna vlákna, která by měla aktuálně běžet a zpracovávat požadavky od uživatelů. Dále periodická kontrola funkce těchto vláken a provádění operací vedoucích k správnému chodu služby. Poslední zodpovědností hlavní služby je zpracovávání požadavků o ukončení běžícího vlákna či naopak spuštění vlákna zastaveného/neexistujícího.



Obrázek 17: Životní cyklus hlavního vlákna

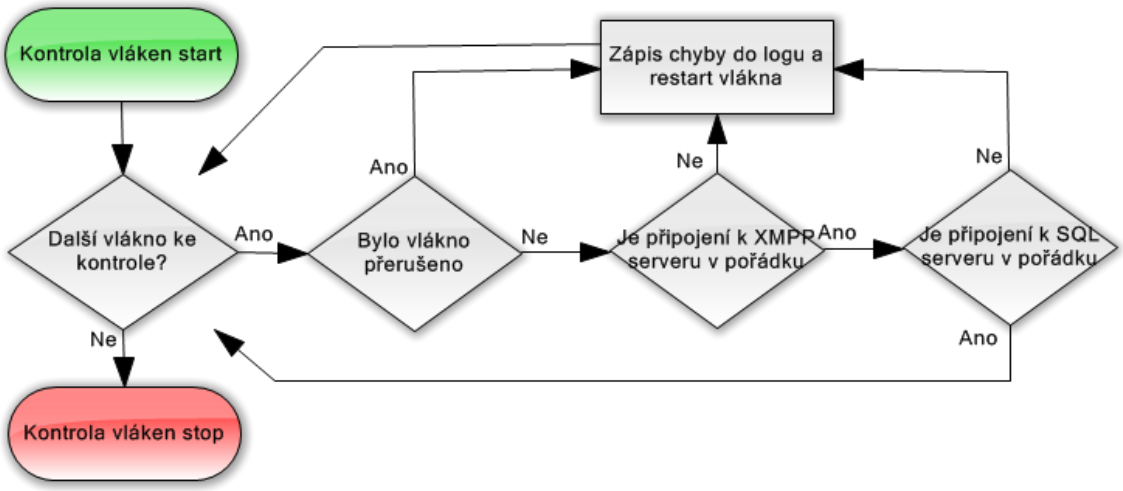
Před spuštěním zmíněných mechanismů dochází k nastavení služby. Tzn. vytvoření objektu pro logování a jeho předání třídě *Logger*, která se stará o zápis do logu (logovat lze do tří výstupních streamů – Konzole, Soubor či Event Logger), vytvoření časovače *System.Timers.Timer*, který v pravidelných intervalech (délka intervalu je nastavitelná pomocí parametru služby) spouští rutiny pro kontrolu a správu vláken, a vytvoření objektu hlavního vlákna *XMPPMainThread*.

Proces inicializace má několik základních kroků (základní princip viz obrázek číslo 18). Nejdříve je vytvořen seznam aktuálně běžících vláken kamer (seznam objektů *XMPPClientThread*). Dále je z databáze XMPP serveru načten seznam všech účtů, které jsou označeny jako vlákna kamery (všechny se nachází ve vybrané skupině, jejíž název je hlavní službě předán jako parametr). Následně je pomocí vložené procedury z databáze samotné služby získán seznam všech vláken, u kterých bylo posledním zpracovaným požadavkem (pouze z variant spuštění/zastavení vlákna) spuštění vlákna. V další fázi jsou procházeny všechny dostupné účty a dochází u nich k vyhodnocení, zda má či nemá být vlákno spuštěno. Jestliže ano, je vytvořen příslušný objekt, zavolána metoda Start (viz kapitola 4.3) a objekt přidán do seznamu běžících vláken. Poslední fází inicializace je pak spuštění kontrolní metody pro každé z běžících vláken. Tato kontrolní metoda pouze čeká předem stanovený čas (jeden ze vstupních parametrů hlavní služby) a pak zkontroluje aktuální stav vlákna vzhledem k připojení k XMPP serveru (ve skutečnosti je čas rozdělen na desetiny a kontrola probíhá cyklicky, aby nebylo nutné čekat celý uvedený čas). Jestliže vlákno nebylo správně spuštěno, dojde k zapsání chyby do logu.



Obrázek 18: Hlavní vlákno – Inicializace

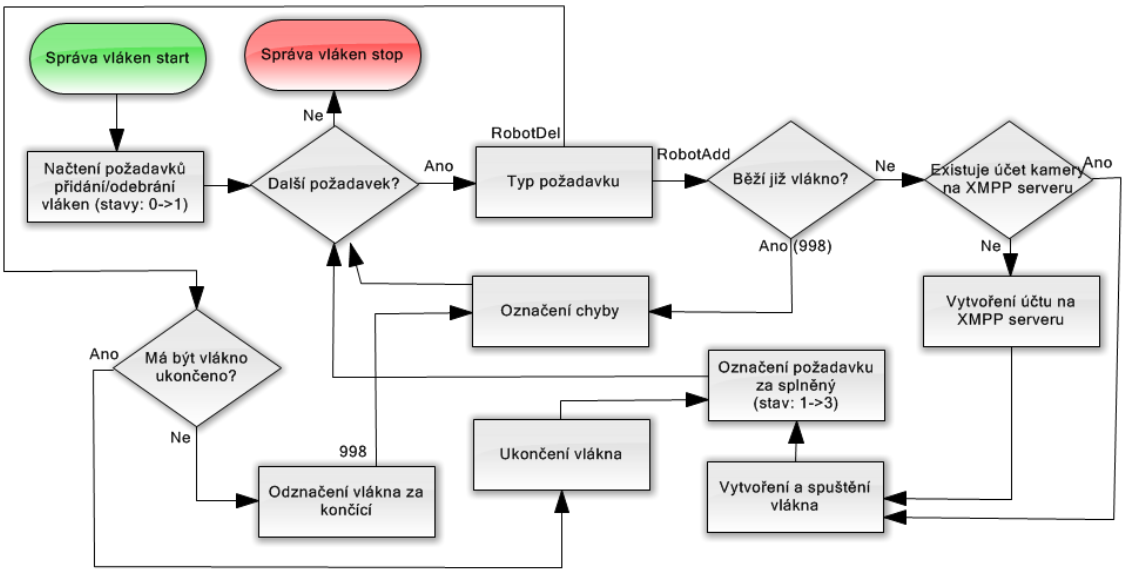
Proces kontroly funkce běžících vláken (viz obrázek číslo 19) prochází postupně všechna běžící vlákna a provádí kontrolu tří základních funkcí vlákna. První je běh samotného vlákna, které je vytvořeno uvnitř třídy *XMPPClientThread*, jeho kontrola je provedena ověřením stavu vlákna pomocí vlastností *ThreadState* a *IsAlive*. Druhou je kontrola připojení ke XMPP serveru, je provedena pomocí vlastnosti *XmppConnectionState* objektu *XmppClientConnection*, který spojení se serverem zajišťuje. Poslední kontrola se týká připojení vlákna kamery k databázi služby. Obsluhování požadavků vlákna probíhá pomocí jednoho „velkého“ připojení k databázi, které umožňuje paralelní zpracování více požadavků najednou, místo několika „malých“ připojení, aby tak nebylo při obsluze stovek vláken aktivních několik tisíc různých malých připojení. Kontrola probíhá tak, že je zjištěno kolik požadavků je v aktuální chvíli vykonáváno a v závislosti na tom je zkontrolován stav připojení pomocí vlastnosti *ConnectionState* objektu *SqlConnection*. Pokud některá z kontrol selže, dochází k zápisu chyby do logu, automatickému zastavení vlákna (zastavení vnitřního vlákna, ukončení spojení s databází i s XMPP serverem), jeho opětovnému spuštění a spuštění kontrolní metody.



Obrázek 19: Hlavní vlákno – Kontrola vláken

Poslední funkcí hlavního vlákna je správa běžících vláken, to znamená obsluha požadavků o přidání a ukončení vlákna (obrázek číslo 20). Ve standardním stavu požadavek o ukončení vlákna odesílá hlavní službě samo vlákno v okamžiku, kdy nemá žádného uživatele, kterého by mělo obsluhovat. Hlavní vlákno takový požadavek převezme a provede nejdříve kontrolu, zda vlákno, které má být ukončeno, opravdu běží (jestliže ne, označí požadavek trvalou chybou). Jestliže vlákno běží, dojde ke kontrole počtu uživatelů jejichž požadavky vlákno vykonává a také vnitřního stavu vlákna

(to se označuje ke smazání společně s odesláním požadavku). Pokud má vlákno nula uživatelů a jeho vnitřní stav je nastaven na „smazat“, dojde k jeho zastavení a označení požadavku na smazání za doručený. Má-li ovšem vlákno přiřazeno alespoň jednoho uživatele, k jeho ukončení nedojde, vnitřní stav je nastaven na „běžící“ a požadavek označen trvalou chybou.



Obrázek 20: Hlavní vlákno – Správa vláken

Požadavek o spuštění vlákna je ve standardním stavu generován WCF službou v okamžiku, kdy vlákno neběží a je mu přiřazen nový uživatel. Před pokusem o spuštění vlákna je zkontrolováno, zda již vlákno neběží, a pokud ano, je požadavek označen trvalou chybou. V opačném případě dojde k vytvoření objektu *XMPPClientThread* s požadovaným jménem robota. Následně musí být na XMPP serveru ověřena existence účtu kamery. Pokud takový účet již existuje, dojde k zavolání metody Start, označení požadavku za doručený a spuštění kontrolní metody běžícího vlákna. Jestliže účet na XMPP serveru neexistuje, je zformulován HTTPS request. Ten je odeslán na XMPP server, který jej zpracuje pomocí rozšíření User Service, a pokusí se vytvořit příslušný XMPP účet (přímé vytvoření není možné, viz kapitola 4.1.7). Jestliže se vytvoření zdaří, dojde k doručení odpovědi ve formátu „<result>OK</result>“, v opačném případě je doručena odpověď „<error>Text chyby</result>“. Jestliže se účet podaří vytvořit, je vlákno spuštěno stejně jako v případě, že by účet existoval již před tím, v opačném případě je požadavek označen opravitelnou chybou, doplněnou o text chyby při vytváření.

4.5 Administrátorské rozhraní

Poslední aplikace, která je v rámci celé služby vytvořena, je webová stránka sloužící k její administraci (ukázka viz obr. č. 21). Je vytvořena pomocí architektury ASP. NET MVC 2 a využívá ke své funkci vytvořenou službu WCF, díky které získává všechna potřebná data a zobrazuje je správci a naopak zasílá jednotlivým vláknům kamer příkazy, které je nutné vykonat. Při první návštěvě webu je dostupná pouze úvodní obrazovka, na které se nachází přihlašovací formulář. Přihlásit se lze dvojím způsobem: účtem a heslem kamery nebo pomocí jména a hesla administrátora.

Obrázek 21: Ukázka administrátorského rozhraní

Jestliže se do systému přihlásí uživatel jménem a heslem kamery, dojde k jeho přesměrování na stránku s názvem „List of Cameras“. Na této stránce je pouze jediná položka s názvem „Available robot Accounts“, v jejímž těle je pouze bare JID přihlášené kamery a obrázek představující její stav na XMPP serveru. Po kliknutí na

zmíněné bare JID, dojde k přesměrování na stránku „Robot Info“, kde se nachází tři položky: „Contacts of Robot“ a v jeho těle všechny kontakty nacházející se v kontakt listu uloženém na XMPP serveru. Dále „Request incoming from users“, kde jsou zobrazeny všechny požadavky odeslané ze strany uživatelů a „Request outcoming from robot“, kde jsou zobrazeny všechny požadavky zpracované robotem a odeslané uživatelům. Obě skupiny položek lze filtrovat podle různých kategorií (podrobnosti později při popisu administrátorského vzhledu stránky).

Jestliže se do systému přihlásí uživatel pomocí administrátorského jména a hesla, dochází k jeho přesměrování, stejně jako v předchozím případě. Administrátor ovšem vidí navíc nabídku „Create new robot account“, která obsahuje formulář pro vložení nového účtu kamery. Do formuláře je nutné vyplnit jméno kamery (pouze localpart XMPP adresy kamery), bare JID klienta (nelze spustit vlákno kamery bez klienta) a případnou uvítací zprávu pro klienta (nepovinná). V nabídce „Available robot Accounts“ pak vidí bare JID všech účtů kamer na XMPP serveru, doplněné o obrázky představující aktuální stav vlákna této kamery vzhledem ke XMPP serveru (k tomuto zobrazení je využito rozšíření serveru Openfire s názvem *Presence Service*, které po odeslání příslušného HTTP requestu zobrazí zmíněnou ikonu). Po kliknutí na kterékoli bare JID kamery dochází k přesměrování na stránku „Robot Info“ (při pokusu o přechod na stránku „Robot Info“, bez předchozího výběru účtu kamery, je uživatel přesměrován na chybovou stránku).

Na této stránce v administrátorské verzi také přibyla jedna nabídka, kterou je „Send request to current robot“ obsahující formulář pro zaslání požadavku na vlákno kamery. První položkou je „Request Type“, kde lze pomocí combo boxu vybrat typ požadavku (odeslání zprávy, změna statusu, přidání kontaktu, odebrání kontaktu, videohovor ze strany kamery, data pro videohovor ze strany klienta či odeslání zprávy se soubory). Po výběru typu požadavku se automaticky doplní zbytek povinných i nepovinných polí, které se požadavku týkají (skrývání a zobrazování polí je realizováno JavaScriptem). Po odeslání požadavku je zobrazeno buď zelené potvrzující hlášení ve spodu formuláře, signalizující odeslání požadavku nebo červený výpis obsahující chybu, která nastala (typicky výpis polí, která nejsou vyplněna či jsou vyplněna chybnou hodnotou).

Pod touto nabídkou je klasicky umístěna nabídka „Contacts of Robot“ se všemi kontakty kamery a pod ní nabídky „Request incoming from users“ a „Request out-

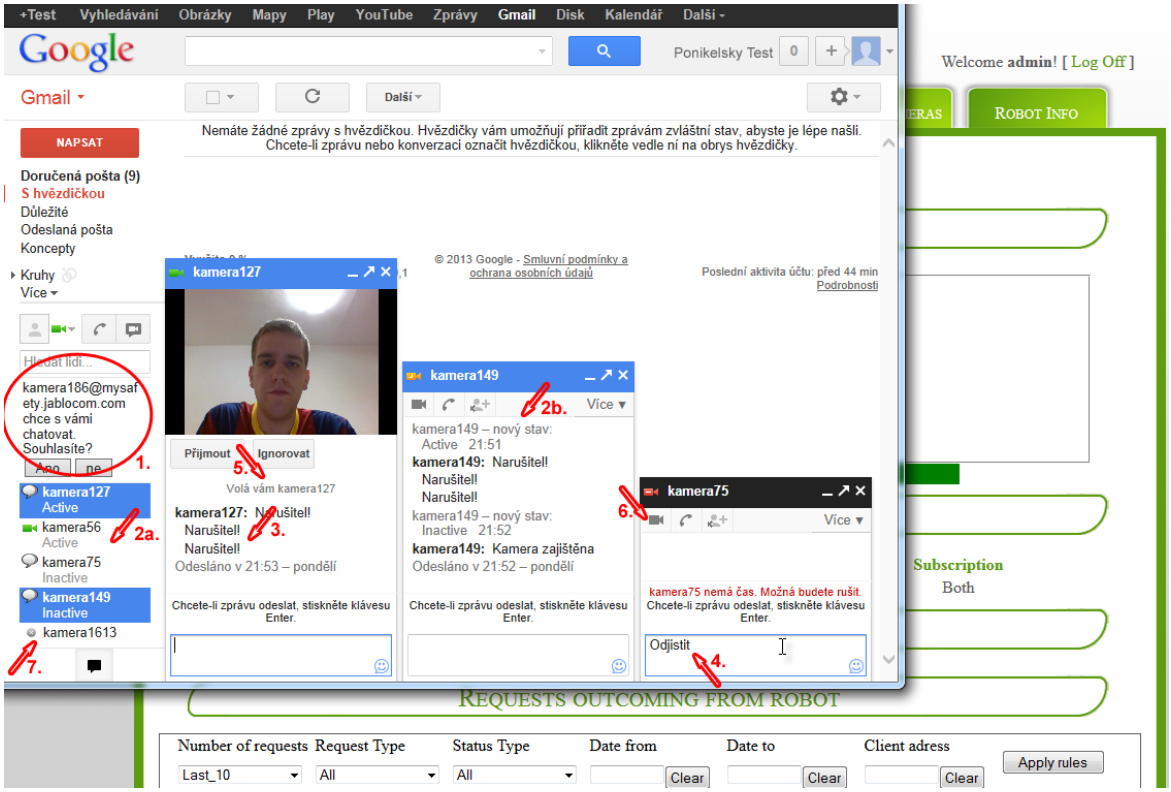
coming from robot“ s požadavky týkajícími se kamery. Libovolnou z těchto nabídek lze složit či rozložit kliknutím na její hlavičku (volba se uchovává pomocí JavaScriptu v cookies prohlížeče, takže přežívá i vypnutí a zapnutí prohlížeče). Nabídky vypisující požadavky od kamery či od klienta lze filtrovat pomocí šesti kritérií (počet požadavků, typ požadavků, stav zpracování požadavků, klient, jehož se požadavek týká, a také časové rozmezí, kdy byl požadavek zapsán do databáze služby). První tři nabídky jsou realizovány výběrem z combo boxu, do pole pro klienta je nutné zadat bare JID klienta nebo ho nechat prázdné, pokud chce uživatel vybrat všechny klienty. Nabídky pro datum lze zapsat ručně ve formátu „měsíc/den/rok hodina:minuta“ nebo k zápisu data použít interaktivní kalendář (realizovaný komponentou *jQuery UI Datepicker* doplněnou o *Timepicker*). Nabídky s položkami se stejně jako seznam bare JID kamer obnovují každých pět sekund (opět realizováno pomocí JavaScriptu, který v periodických intervalech volá specifický controller, který mu vrací aktuální hodnoty zformátované do výsledné podoby). Tzn. po stisknutí tlačítka „Apply rules“, nedochází k okamžité změně všech zobrazených požadavků (pouze se změní řetězec, který bude při dalším obnovení nabídky odeslán controlleru). Ukázka stránky „Robot info“ je zobrazena na obrázku číslo 21.

4.6 Uživatelské rozhraní

Celkový výsledek diplomové práce z pohledu uživatele služby bude nejlépe vidět na ukázce interakce kamery se skutečným XMPP klientem. Na obrázku číslo 22 je vidět ukázka XMPP klienta Google Chat. Prvním krokem při interakci s uživatelem bude přidání jeho adresy do kontaktů kamery, zaslání požadavku o povolení stavové informace z jeho strany (viz 1. bod na obrázku č. 22) a také povolení stavové informace ze strany kamery. Následně bude moci uživatel vidět aktuální stav kamery a bude průběžně informován o jeho změně (viz 2a. a 2b.). Jestliže dojde k nějaké naléhavé situaci na straně kamery, uživatel bude informován prostřednictvím zaslané zprávy (viz 3.).

Naopak, pokud bude chtít uživatel provést změnu na některé z kamer v jeho kontakt listu, provede ji odesláním zprávy ze svého XMPP klienta (viz 4.). Jestliže uživatel používá některého z klientů společnosti Google inc. podporujícího funkci videohovorů, existuje také možnost, že samotná kamera inicializuje videohovor (viz 5.). Naopak videohovor si může vyžádat i klient a to stiknutím tlačítka pro videohovor (viz 6.). Nebude-li chtít klient nadále získávat informace z kamery, dojde k jeho odebrání z kon-

takt listu kamery (dále nebude moci nadále vidět její stav – viz 7.). Je důležité dodat, že Google Chat je pouze jedním z mnoha XMPP klientů. Google chat navíc nepodporuje žádnou technologii přenosu souborů, tu tak mohou využít například uživatelé XMPP klientů Miranda, Pidgin, Psi, Spark apod.



Obrázek 22: Ukázka ovládání služby – Klient Google chat

5 Závěr

Cílem této diplomové práce bylo umožnit zákazníkům společnosti Jablocom s. r. o. ovládání zabezpečovací kamery EYE-02, která slouží k ochraně jejich zdraví či majetku, pomocí XMPP klienta, kterého používají ke komunikaci se svými spolupracovníky či blízkými – tzn. bez nutnosti instalovat nového XMPP klienta či rozšiřovat stávajícího o požadované moduly.

Prvním úkolem nutným k dosažení tohoto cíle byl výběr XMPP serveru. Po instalaci několika vybraných serverů a prověření rozsahu jejich konfigurace, byl vybrán server Openfire, který v konkurenci zbylých serverů (Ejabberd, Prosody, Tigase ...) poskytuje nejlepší možnosti v oblasti nastavení (např. nastavení bezpečnostních nástrojů protokolu XMPP, které umožní zabránit odposlechnutí či modifikaci zprávy nebo napadení samotného serveru), podpory rozšiřujících standardů protokolu XMPP a dalších technologií, nutných například pro přenos souborů či videohovorů, pohodlnosti uživatelské obsluhy a využití stávajících programových prostředků společnosti Jablocom.

Dalším postupem pak byl návrh databáze služby, která tvoří prostředníka mezi vyvinutou službou a existující infrastrukturou společnosti a dokáže pojmout všechny důležité údaje. Součástí této databáze je pak WCF služba, jejímž prostřednictvím je realizováno zadávání příkazů do této databáze a naopak sběr příchozích dat ze strany klienta, bez nutnosti porozumění struktuře databáze.

Dále pak byla vytvořena služba, která umožňuje dynamickou správu účtů kamer na XMPP serveru (přidávání/odebírání účtů kamer, přidávání/odebírání kontaktů těchto účtů) a neustálou kontrolu jejich chodu. Prostřednictvím těchto účtů jsou pak odesílány uživatelské zprávy z kamer EYE-02 na IM klienty zákazníků. Součástí toho řešení byl i pokus o ověření doručitelnosti těchto zpráv, tedy implementaci rozšíření XMPP protokolu pro tyto účely sloužících. Bohužel schopnost ověření doručení těchto zpráv ztroskotala na omezené podpoře těchto rozšíření externími XMPP klienty a servery.

Pomocí této služby je také možné kameru ovládat a to zadáváním příkazů do konverzačního okna kamery. Služba tyto povely přenáší dále do informační databáze společnosti Jablocom, která pak povely odesílá přímo na kameru. Ta reaguje změnou svého stavu, kterou může uživatel pozorovat prostřednictvím změny stavu účtu kamery či jinou interakcí (příjem zprávy apod.).

Posledním bodem zadání pak bylo prověřit možnosti přenosu videohovorů prokolem XMPP. V této oblasti byla služba přizpůsobena upravené verzi standardu společnosti Google, která byla prioritní díky své široké základně zákazníků. Služba je teoreticky připravena videohovory inicializovat jak ze své strany (kamera začne volat na XMPP účet zákaznka), tak také reagovat na volání samotného zákazníka. Bohužel v současné době společnost Jablocom přenáší video k zákazníkům pomocí protokolu, který není ve VOIP protokolu XMPP podporován, tudíž k plné funkcionalitě této části je nutné počkat na jeho konverzi.

Tímto byly splněny všechny body zadání diplomové práce. Nad rámec zadání práce pak byla vytvořena možnost odesílání souborů na vybrané XMPP klienty zákazníků, pomocí rozšíření SOCKS5 Bytestream protokolu XMPP. Také byla vytvořena webová stránka, sloužící jako administrativní nástroj, jehož prostřednictvím je možné odesílat povely účtům kamer a také vytvářet účty nové. Tato webová stránka zároveň slouží pro možnosti testování funkčnosti celé služby zaměstnanci firmy Jablocom, kteří si prostřednictvím tohoto webu mohou otestovat funkci služby se všemi zájmovými externími servery (např. gmail.com, jabbim.cz, jabber.org, jabber-server.de apod.).

Během vývoje služby byla vyvíjena maximální snaha o upravitelnost výsledného kódu samotnými zaměstnanci společnosti, pod kterou si lze představit přehledné členění kódu a velké množství komentářů (zejména komentáře funkcí, ze kterých lze vytvořit výslednou dokumentaci), ladící chybová hlášení a možnost jejich logování do souboru či prostřednictvím nástroje Event Viewer. Služba je pak složena z přibližně čtyř tisíc řádků zdrojového kódu (nejsou započítány komentáře ani prázdné řádky).

Možností jak tuto aplikaci rozšířit existuje celá řada. Protokol XMPP obsahuje celou řadu rozšíření, které by bylo možné do služby doimplementovat a zajistit tak větší komfort pro zákazníky. Problém těchto rozšíření je ovšem častá nekompatibilita s existujícími XMPP klienty a XMPP servery, takže jednou z dalších možností rozšíření je i vytvoření vlastního grafického XMPP klienta, který by byl nabídnut všem zákazníkům společnosti Jablocom a podporoval požadovaná rozšíření, či vytvoření vlastního XMPP serveru.

Seznam použité literatury

[1] B MEDIA SOLUTIONS. Jablocom [online]. Jablonec nad Nisou: Jablocom, © 2011 [cit. 2013-03-17]. Dostupné z: http://www.jablocom.cz/.

[2] XMPP Standards Foundation [online]. Denver (Colorado): XMPP Standards Foundation, [August 21, 2012] [cit. 2012-11-09]. Dostupné z: http://xmpp.org/.

[3] NEUBAUER, Peter. Social Computing or Let the bots talk!: What is XMPP?. In: Internetdagarna [online]. [31 augusti 2009] [cit. 2012-11-10]. Dostupné z: www.internetdagarna.se/track/media/social-computing-or-let-the-bots-talk.

[4] Peter SAINT-ANDRE. *Extensible Messaging and Presence Protocol (XMPP): Address Format: RFC 6122*. [Fremont (California)]: Internet Engineering Task Force (IETF), March 2011. ISSN 2070-1721. Dostupné z: http://xmpp.org/rfcs/rfc6122.html.

[5] Peter SAINT-ANDRE. *Extensible Messaging and Presence Protocol (XMPP): Core: RFC 6120*. [Fremont (California)]: Internet Engineering Task Force (IETF), March 2011. ISSN 2070-1721. Dostupné z: http://xmpp.org/rfcs/rfc6120.html.

[6] Peter SAINT-ANDRE. *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence: RFC 6121.* [Fremont (California)]: Internet Engineering Task Force (IETF), March 2011. ISSN 2070-1721. Dostupné z: http://xmpp.org/rfcs/rfc6121.html.

[7] JIVE SOFTWARE. Ignite Realtime: a real time collaboration community site [online]. © 2013, Mar 3, 2013 3:59 PM [cit. 2013-04-15]. Dostupné z: http://www.igniterealtime.org/.

[8] SHCHEPIN, Alexey. Ejabberd Community Site: the Erlang Jabber/XMPP daemon [online]. © 2002, Fri, 2012-05-04 15:06 [cit. 2013-04-01]. Dostupné z: http://www.ejabberd.im/.

[9] Jabberd2: JabberD XMPP Server. GITHUB, Inc. GitHub: Build software better, together. [online]. © 2013 [cit. 2013-04-10]. Dostupné z: https://github.com/jabberd2/jabberd2.

[10] TIGASE INC. Tigase.org: Open Source and Free XMPP/Jabber Software [online]. © 2012, Thu, 2013-03-28 16:10 [cit. 2013-04-08]. Dostupné z: http://www.tigase.org/.

[11] PROSODY TEAM. Prosody: A study in simplicity [online]. [2012], 2012-09-28 [cit. 2013-04-05]. Dostupné z: http://prosody.im/.

[12] AgsXMPP SDK. AG-SOFTWARE. XMPP Components and Competence [online]. © 2013 [cit. 2013-04-11]. Dostupné z: http://www.ag-software.net/agsxmpp-sdk/.

[13] MILLER, Jeremie, DJ ADAMS a Peter SAINT-ANDRE. XEP-0022: Message Events. XMPP STANDARDS FOUNDATION. XMPP Standards Foundation [online]. Version: 1.4. © 1999 – 2013, 2009-05-27 [cit. 2013-05-08]. Dostupné z: http://xmpp.org/extensions/xep-0022.html.

[14] MILLER, Matthew a Peter SAINT-ANDRE. XEP-0079: Advanced Message Processing. XMPP STANDARDS FOUNDATION. XMPP Standards Foundation [online]. Version: 1.2. © 1999 – 2013, 2005-11-30 [cit. 2013-05-08]. Dostupné z: http://xmpp.org/extensions/xep-0079.html.

[15] HILDEBRAND, Joe, Peter SAINT-ANDRE. XEP-0184: Message Delivery Receipts. XMPP STANDARDS FOUNDATION. XMPP Standards Foundation [online]. Version: 1.2. © 1999 – 2013, 2011-03-01 [cit. 2013-05-08]. Dostupné z: http://xmpp.org/extensions/xep-0184.html.

[16] Google Talk Call Signaling. GOOGLE, Inc. Google Developers: Google Talk for Developers [online]. Version 2.0. [2011], March 23, 2012 [cit. 2013-05-08]. Dostupné z: https://developers.google.com/talk/call_signaling.

[17] LUDWIG, Scott, Peter SAINT-ANDRE, Sean EGAN, Robert MCQUEEN a Diana CIONOIU. XEP-0167: Jingle RTP Sessions. XMPP STANDARDS FOUNDATION. XMPP Standards Foundation [online]. Version: 1.1. © 1999 – 2013, 2009-12-23 [cit. 2013-05-11]. Dostupné z: http://xmpp.org/extensions/xep-0167.html.

[18] ATWOOD, Jeff a Joel SPOLSKY. Stack Overflow [online]. [New York], [2008], rev 2013.4.10.630 [cit. 2013-04-11]. Dostupné z: http://stackoverflow.com/

[19] MICROSOFT. MSDN Library [online]. [Redmont], © 2013 [cit. 2013-04-11]. Dostupné z: http://msdn.microsoft.com/en-us/library

[20] MAUNDER, Chris et al. CodeProject: For those who code [online]. © 1999-2013, Last Updated 11 Apr 2013 [cit. 2013-04-11].
Dostupné z: http://www.codeproject.com/

[21] THE JQUERY FOUNDATION. JQuery API Documentation [online].
© 2013 [cit. 2013-04-11]. Dostupné z: http://api.jquery.com/

[22] RICHARDSON, Trent. Adding a Timepicker to jQuery UI Datepicker.
RICHARDSON, Trent. Trent Richardson: practical web design & development [online]. Version 1.2.1. © 2013, Last updated on 04/06/2013 [cit. 2013-04-11].
Dostupné z: http://trentrichardson.com/examples/timepicker/

[23] HARTL, Klaus. Jquery-cookie: A simple, lightweight jQuery plugin for reading, writing and deleting cookies. GITHUB, Inc. GitHub: Build software better, together. [online]. © 2013 [cit. 2013-04-11].
Dostupné z: https://github.com/carhartl/jquery-cookie

Příloha A – Obsah CD

Diplomová práce ve formátu PDF

Skripty pro vytvoření databáze služby

WCF služba pro práci s databází (Projekt – Visual Studio 2010)

Vytvořená XMPP služba (Projekt – Visual Studio 2010)

Administrační webová stránka (Projekt – Visual Studio 2010)

Nástroj pro úpravu konfiguračního souboru služby (Projekt – Visual Studio 2010)

Instalátor serveru OpenFire verze 3.8.1

Licence serveru OpenFire

Knihovna agsXMPP použitá při vývoji

Licence knihovny agsXMPP